(FO 1334 2 US) /ed

APPL NO. 09/299,785
GAU: 2782

日 本 国 特 許 庁
PATENT OFFICE
JAPANESE GOVERNMENT

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application: １９９８年 ４月２２日

出願番号
Application Number: 平成１０年特許願第１１１６８１号

出願人
Applicant (s): キヤノン株式会社

CERTIFIED COPY OF
PRIORITY DOCUMENT

１９９９年 ３月１９日

特許庁長官
Commissioner,
Patent Office 伊佐山建志

出証番号 出証特平１１－３０１３０１０

【書類名】　特許願

【整理番号】　3740013

【提出日】　平成10年 4月22日

【あて先】　特許庁長官殿

【国際特許分類】　H04L 12/00

【発明の名称】　データ通信システム、方法及び装置並びに記憶媒体

【請求項の数】　14

【発明者】

【住所又は居所】　東京都大田区下丸子３丁目３０番２号　キヤノン株式会社内

【氏名】　大西　慎二

【発明者】

【住所又は居所】　東京都大田区下丸子３丁目３０番２号　キヤノン株式会社内

【氏名】　波多江　真一

【発明者】

【住所又は居所】　東京都大田区下丸子３丁目３０番２号　キヤノン株式会社内

【氏名】　新井田　光央

【発明者】

【住所又は居所】　東京都大田区下丸子３丁目３０番２号　キヤノン株式会社内

【氏名】　小林　崇史

【特許出願人】

【識別番号】　000001007

【氏名又は名称】　キヤノン株式会社

【代理人】

【識別番号】　100090273

【弁理士】

【氏名又は名称】　國分　孝悦

【電話番号】　03-3590-8901

【手数料の表示】

【予納台帳番号】　035493

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【包括委任状番号】　9705348

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　データ通信システム、方法及び装置並びに記憶媒体

【特許請求の範囲】

【請求項１】　所定の機器に固有の固有情報と、送信機器と１つ以上の受信機器との間の論理的な接続を示すコネクション情報とを用いて、情報データの送受信を行うデータ通信システムにおいて、

上記１つ以上の受信機器の受信能力のうち、最も低い受信能力に従って上記情報データの送受信を行うことを特徴とするデータ通信システム。

【請求項２】　上記所定の機器は、上記コネクション情報を設定する管理機器であることを特徴とする請求項１に記載のデータ通信システム。

【請求項３】　上記管理機器は、上記情報データの送信に先立って、上記固有情報と上記コネクション情報とを上記送信機器と上記１つ以上の受信機器とに通知することを特徴とする請求項１または２に記載のデータ通信システム。

【請求項４】　上記送信機器と上記１つ以上の受信機器とは、上記管理機器を介することなく、上記固有情報と上記コネクション情報とを用いて上記情報データの送受信を行うことを特徴とする請求項２または３に記載のデータ通信システム。

【請求項５】　上記データ通信システムの接続構成が初期化された後、上記送信機器と上記１つ以上の受信機器とは、上記管理機器を介することなく上記情報データの送受信を再開することを特徴とする請求項１～４の何れか１項に記載のデータ通信システム。

【請求項６】　上記データ通信システムの接続構成が初期化された後、上記送信機器と上記１つ以上の受信機器とは、上記固有情報と上記コネクション情報とを用いて上記情報データの送受信を再開することを特徴とする請求項１～５の何れか１項に記載のデータ通信システム。

【請求項７】　上記受信機器の受信能力は、上記受信機器の具備する受信バッファのサイズを含むことを特徴とする請求項１～６の何れか１項に記載のデータ通信システム。

【請求項８】　上記管理機器は、上記送信機器から送信される情報データを

受信する受信機器の数を、上記コネクション情報と共に管理することを特徴とする請求項２または３に記載のデータ通信システム。

【請求項９】　上記情報データは、特定の受信機器を指定しない情報と共に送信されることを特徴とする請求項１～８の何れか１項に記載のデータ通信システム。

【請求項１０】　上記固有情報と上記コネクション情報とは、上記データ通信システムの接続構成が初期化されても変化しないことを特徴とする請求項１～９の何れか１項に記載のデータ通信システム。

【請求項１１】　所定の機器に固有の固有情報と、送信機器と１つ以上の受信機器との間の論理的な接続を示すコネクション情報とを用いて、情報データの送受信を行うデータ通信システムに適用可能なデータ通信方法において、

上記１つ以上の受信機器の受信能力のうち、最も低い受信能力に従って上記情報データの送受信を実行させることを特徴とするデータ通信方法。

【請求項１２】　所定の機器に固有の固有情報と、１つ以上の受信機器との間の論理的な接続を示すコネクション情報とを用いて情報データの送受信を行う通信手段と、

上記１つ以上の受信機器の受信能力のうち、最も低い受信能力に従って上記通信手段の動作を制御する制御手段とを具備することを特徴とするデータ通信装置。

【請求項１３】　請求項１１に記載のデータ通信方法の手順をコンピュータに実行させるためのプログラムを格納したことを特徴とする記憶媒体。

【請求項１４】　請求項１２に記載の各手段としてコンピュータを機能させるためのプログラムを格納したことを特徴とする記憶媒体。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明はデータ通信システム、方法及び装置並びに記憶媒体に関し、特に、制御信号とデータを混在させて通信することが可能なデータ通信バスを用いて複数電子機器（以下、機器）間を接続して、各機器間でデータ通信を行うシステムを

所有する装置に用いて好適なものである。

【０００２】

【従来の技術】

パソコン周辺機器の中で、最も利用頻度が高いのはハードディスクやプリンタであり、これらの周辺装置は小型コンピュータ用汎用型インターフェイスで代表的なデジタルインターフェイス（以下、デジタルI/F)であるSCSI等を介してパソコン間との接続がなされ、データ通信が行われている。

【０００３】

また、デジタルカメラやデジタルビデオカメラといった記録再生装置もパソコン（以下、PC）への入力手段として用いられる周辺装置の１つであり、近年、デジタルカメラやビデオカメラで撮影した静止画や動画といった映像をパソコンPCへ取り込み、ハードディスクに記憶したり、またはパソコンPCで編集した後、プリンタでカラープリントするといった分野の技術が進んでおり、ユーザーも増えている。

【０００４】

取り込んだ画像データをパソコンPCからプリンタやハードディスクへ出力する際などに、上記のSCSI等を経由してデータ通信がされるものであり、そのようなときに画像データのようなデータ量の多い情報を送るためにも、こういったデジタルI/F には転送データレートの高い、かつ汎用性のあるものが必要とされる。

【０００５】

図２１に、従来の例としてデジタルカメラ、パソコンPC及びプリンタを接続したときのブロック図を示す。

図２１において、101 はデジタルカメラ、102 はパソコン(PC)、103 はプリンタである。さらに、104 はデジタルカメラの記録部であるメモリ、105 は画像データの復号化回路、106 は画像処理部、107 はD/A コンバータ、108 は表示部であるEVF 、109 はデジタルカメラのデジタルI/0 部、110 はパソコンPCのデジタルカメラとのデジタルI/0 部、111 はキーボードやマウスなどの操作部である。

【０００６】

112 は画像データの復号化回路、113 はディスプレイ、114 はハードディスク

装置、115 はRAM 等のメモリ、116 は演算処理部のMPU 、117 はPCI バス、118 はデジタルI/F のSCSIインタフェース（ボード）、119 はパソコンPCとSCSIケーブルで繋がったプリンタのSCSIインターフェイス、120 はメモリ、121 はプリンタヘッド、122 はプリンタ制御部のプリンタコントローラ、123 はドライバである。

【０００７】

先ず、デジタルカメラで撮像した画像をパソコンPCに取り込み、またパソコンPCからプリンタへ出力するときの手順の説明を行う。デジタルカメラ101 のメモリ104 に記憶されている画像データが読み出されると、上記読み出された画像データのうちの一方は復号化回路105 で復号化され、画像処理回路106 で表示するための画像処理がなされ、D/A コンバータ107 を経て、EVF108で表示される。

【０００８】

また一方では、外部出力するためにデジタルI/O 部109 から、ケーブルを伝わってPC102 のデジタルI/O 部110 へ至る。PC102 内では、PCI バス117 を相互伝送のバスとして、デジタルI/O 部110 から入力した画像データは、記憶する場合はハードディスク114 で記憶され、表示する場合は復号化回路112 で復号化された後、メモリ115 で表示画像としてメモリされて、ディスプレイ113 でアナログ信号に変換されてから表示される。パソコンPC102 での編集時等の操作入力は操作部111 から行い、パソコンPC102 全体の処理はMPU116で行う。

【０００９】

また、画像をプリント出力する際は、パソコンPC102 内のSCSIインターフェイスボード118 から画像データをSCSIケーブルを介して伝送し、プリンタ103 側のSCSIインターフェイス119 で受信し、メモリ120 でプリント画像として形成され、プリンタコントローラ122 の制御でプリンタヘッド121 とドライバ123 が動作して、メモリ120 から読み出したプリント画像データをプリントする。

【００１０】

以上が、従来の画像データをパソコンPC取り込み、またはプリントするまでの手順である。このように、従来はホストであるパソコンPCにそれぞれの機器が接続され、パソコンPCを介してから、記録再生装置で撮像した画像データをプリン

トしている。

【００１１】

また、ディジタルVTR 、TV、チューナなどのAV機器や、パーソナルコンピュータ（以下、PCと称する）等をIEEEP1394 シリアルバス（以下、1394と称する）を用いて相互に接続し、これらの間においてディジタルビデオ信号、ディジタルオーディオ信号などを送受信するデータ通信システムが提案されている。

【００１２】

これらのシステムにおいては、リアルタイムにデータ転送することが重要となるため、いわゆる同期通信（以下、Isochronous 通信と称する）によって、データ通信を行なっている。この場合には、データ転送のリアルタイム性は保証されるが、通信が確実に実行されることは保証されない。

【００１３】

【発明が解決しようとする課題】

しかしながら、上記従来例で挙げたデジタルインターフェイスの問題点として、SCSIには転送データレートの低いものや、パラレル通信のためケーブルが太いもの、接続される周辺機器の種類や数、接続方式などにも制限があり、多くの面での不便利性も指摘されている。

【００１４】

また、従来の1394通信の場合には、同期通信を行なうため、通信が確実に実行されることは保証されない。したがって、確実にデータ転送を行ないたい場合には、従来の1394 Isochronous通信を使用することはできなかった。

【００１５】

また、従来の1394 Isochronous通信では、通信帯域に空きがある場合にも、通信の総数が64に制限される。このため、通信帯域をあまり要求しないような通信を多数行ないたい場合には、従来の1394 Isochronous通信を使用することはできないといった問題点があった。

【００１６】

また、従来の1394通信方式では、データ転送の間に、バスリセットやエラーによる、データ転送の中断が生じることが考えられる。この場合、従来の1394通信

方式では、どのようなデータ内容が失われたのかを知ることができない問題があった。そのため、従来の1394通信方式では、該データ転送中断からの復帰を行なうためには、非常に繁雑な通信手順を踏むことを要求されるという問題点があった。

【００１７】

本発明は、上記問題点を解決するためになされたもので、従来の通信方式の不利便性を解決し、簡便にかつ高速にデータを転送できるようにするとともに、確実にデータ転送を実行できるようにすることを第１の目的とする。

また、本発明は、通信帯域をあまり使用しない場合に、多数の通信を同時に行なえるようにすることを第２の目的とする。

また、本発明は、データ転送中断により失われたデータを容易に検出することが可能で、上記データ転送中断からの復帰を、確実に、かつ簡単に行なえるようにすることを第３の目的とする。

また、本発明は、情報データの転送を管理する複数のコントロールノードがネットワーク上に存在する場合に、個々のコントロールノードにより管理される通信を識別できる機能を提供することを第４の目的とする。

さらに、本発明は、同じ情報データを１つのソースノードから複数のデスティネーションノードに対して転送することのできる機能を提供することを第４の目的とする。

【００１８】

【課題を解決するための手段】

従来抱えている問題を解決するために、本発明は、従来からあるデジタルI/Fの問題点を極力解消した、各デジタル機器に統一されて搭載されるような汎用型デジタルI/F(例えばIEEE1394-1995 ハイパフォーマンス・シリアルバス）を用いて、パソコンPCやプリンタ、その他周辺装置、またデジタルカメラやデジタルVTR の記録再生装置等をネットワーク構成で接続したときの機器間データ通信を実現し、記録再生装置からビデオデータ等のパソコンPCへの取り込み、また、映像データをプリンタへ直接転送しプリントなどを実現する。このようなネットワークにおいて、各種のデーターをAsynchronousトランザクションによりそれぞれの

データを複数に分割して伝送するプロトコルを提供するものである。

ペイロード内に、バスリセット等によっても変化しない、ソースノード固有のノード情報とソースノードが管理するID番号を組み合わせた識別データを付加する。

コントローラノードは、ソースに対して論理的に接続されたデスティネーション数を通知する。

コントローラノードはデスティネーションノードが複数存在する場合は、各デスティネーションノードが持つバッファサイズの中で最小のバッファサイズをソースノードに通知する。

ソースノードは、コントローラノードから通知されたデスティネーションノードのバッファサイズごとに該デスティネーションからの受信確認応答パケットをまって、次のデータパケットを送信する。

ソースノードからの送信データー終了を示すパケットに対してそれぞれのデスティネーションは、受信確認応答パケットを返すようにしている。

【００１９】

本発明のデータ通信システムは、所定の機器に固有の固有情報と、送信機器と１つ以上の受信機器との間の論理的な接続を示すコネクション情報とを用いて、情報データの送受信を行うデータ通信システムにおいて、上記１つ以上の受信機器の受信能力のうち、最も低い受信能力に従って上記情報データの送受信を行うことを特徴としている。

また、本発明のデータ通信システムの他の特徴とするところは、上記所定の機器は、上記コネクション情報を設定する管理機器であることを特徴としている。

また、本発明のデータ通信システムのその他の特徴とするところは、上記管理機器は、上記情報データの送信に先立って、上記固有情報と上記コネクション情報とを上記送信機器と上記１つ以上の受信機器とに通知することを特徴としている。

また、本発明のデータ通信システムのその他の特徴とするところは、上記送信機器と上記１つ以上の受信機器とは、上記管理機器を介することなく、上記固有情報と上記コネクション情報とを用いて上記情報データの送受信を行うことを特

徴としている。

また、本発明のデータ通信システムのその他の特徴とするところは、上記データ通信システムの接続構成が初期化された後、上記送信機器と上記１つ以上の受信機器とは、上記管理機器を介することなく上記情報データの送受信を再開することを特徴としている。

また、本発明のデータ通信システムのその他の特徴とするところは、上記データ通信システムの接続構成が初期化された後、上記送信機器と上記１つ以上の受信機器とは、上記固有情報と上記コネクション情報とを用いて上記情報データの送受信を再開することを特徴としている。

また、本発明のデータ通信システムのその他の特徴とするところは、上記受信機器の受信能力は、上記受信機器の具備する受信バッファのサイズを含むことを特徴としている。

また、本発明のデータ通信システムのその他の特徴とするところは、上記管理機器は、上記送信機器から送信される情報データを受信する受信機器の数を、上記コネクション情報と共に管理することを特徴としている。

また、本発明のデータ通信システムのその他の特徴とするところは、上記情報データは、特定の受信機器を指定しない情報と共に送信されることを特徴としている。

また、本発明のデータ通信システムのその他の特徴とするところは、上記固有情報と上記コネクション情報とは、上記データ通信システムの接続構成が初期化されても変化しないことを特徴としている。

【００２０】

本発明のデータ通信方法は、所定の機器に固有の固有情報と、送信機器と１つ以上の受信機器との間の論理的な接続を示すコネクション情報とを用いて、情報データの送受信を行うデータ通信システムに適用可能なデータ通信方法において、上記１つ以上の受信機器の受信能力のうち、最も低い受信能力に従って上記情報データの送受信を実行させることを特徴としている。

【００２１】

本発明のデータ通信装置は、所定の機器に固有の固有情報と、１つ以上の受信

機器との間の論理的な接続を示すコネクション情報とを用いて情報データの送受信を行う通信手段と、上記１つ以上の受信機器の受信能力のうち、最も低い受信能力に従って上記通信手段の動作を制御する制御手段とを具備することを特徴としている。

【００２２】

本発明の記憶媒体は、上記データ通信方法の手順をコンピュータに実行させるためのプログラムを格納したことを特徴としている。

また、本発明の記憶媒体の他の特徴とするところは、上記各手段としてコンピュータを機能させるためのプログラムを格納したことを特徴としている。

【００２３】

【作用】

本発明は上記技術手段を有するので、コントローラノードによって設定されたコネクション識別データを使用してソース、デスティネーション間でデータ転送を行うことにより、バスリセットが発生して各ノードのノードIDが変化してもデータ転送を簡単な手順で再開することができる。

また、複数のデスティネーションが異なるサイズのデータバッファを持つ場合には、コントローラノードがその中の最小のバッファサイズをソースに通知することにより、各デスティネーションの持つバッファサイズに関係なく、ソースは上記通知されたバッファサイズ以下のサイズ単位でデータ転送を行うことにより、通信を確実に実行することが可能となる。

【００２４】

【発明の実施の形態】

以下、図面を参照しながら、本発明のデータ通信システム、方法及び装置並びに記憶媒体の実施の形態について説明する。

図１において、10はcomputerであり、12は演算処理装置(MPU)、14は第１の1394インターフェイス、16はキーボードなど第１の操作部、18は第１のデコーダ、20はCRT ディスプレイなどの表示装置、22はハードディスク、24は第１のメモリであり本発明に係るcomputer10の内部メモリ、26はPCI バスなどのコンピュータ内部バスである。

【００２５】

また、28はVCR であり、30は撮像光学系、32はアナログーデジタル(A/D) 変換器、34はビデオ処理部、36は圧縮伸長回路、38は第１のメモリ、40は第２のメモリ、42は第１のデータセレクタ、44は第２の1394インターフェイス、46は第１のメモリ制御回路、48は第２のメモリ制御回路、50はシステムコントローラ、52は第２の操作部、54はファインダ、56はD/A 変換器、58は記録部である。

【００２６】

さらに、60はプリンタであり、62は第３の1394インターフェイス、64は第２のデータセレクタ、66は第３の操作部、68はプリンタコントローラ、70は第２のデコーダ、72は第３のメモリ、74は画像処理部、76はドライバ、78はプリンタヘッドである。

【００２７】

computer10と、VCR28 、及び、プリンタ60とは、第１から第３の1394インターフェイス14,44,62によって1394シリアルバスのノードを構成するとともに、該第１から第３の1394インターフェイス14,44,62を介して相互に接続されており、データの授受や、コマンドによるコントロール等が可能になっている。

【００２８】

本実施の形態では、例えば、computer10は、1394シリアルバス上における、画像信号送受信のコントローラとして動作する。本実施の形態に係るcomputer10においては、例えば、PCI バスなどのコンピュータ内部バス26によって、演算処理装置12と、1394インターフェイス14、キーボード16、デコーダ18、CRT ディスプレイ20、ハードディスク22、内部メモリ24などの、内部の各デバイスとが相互に接続されている。

【００２９】

演算処理装置(MPU)12 は、ハードディスク22に記録されているソフトウェアを実行するとともに、様々なデータを内部メモリ24に移動させる。また、演算処理装置(MPU)12 は、PCI バス26によって接続されている各デバイスの、調停動作なども合わせて行なう。

【００３０】

1394インターフェイス14は、1394シリアルバス上に転送される画像信号を受信するとともに、ハードディスク22に記録されている画像信号や、内部メモリ24に記憶される画像信号を送信する。また、1394インターフェイス14は、1394シリアルバス上に接続された他の機器に対するコマンドデータを送信する。また、1394インターフェイス14は、1394シリアルバス上に転送される信号を他の1394ノードに転送する。

【００３１】

操作者は、キーボード16などの操作部を通じて、MPU12 に、ハードディスク22に記録されているソフトウェアを実行させる。該ソフトウェア等の情報は、CRT ディスプレイなどの表示装置20によって、操作者に提示される。デコーダ18は、上記のソフトウェアを通じて、1394シリアルバス上から受信した画像信号をデコードする。デコードされた画像信号も、また、CRT ディスプレイなどの表示装置20によって、操作者に提示される。

【００３２】

本実施の形態では、例えば、VCR28 は、画像信号の入力装置として動作する。撮像光学系30から入力された映像の輝度信号（Ｙ）と色差信号（Ｃ）は各々A/D 変換器32にてディジタルデータに変換される。上記ディジタルデータは、ビデオ処理部34にて多重化される。その後、圧縮伸長回路36にて該画像情報のデータ量を圧縮する。

【００３３】

一般に、YC独立に該圧縮処理回路を備えているが、ここでは説明の簡略化の為にYC時間分割での圧縮処理の例を示す。次に、上記画像データを伝送路誤りに強くする目的でシャフリング処理を施す。この処理の目的は連続的な符号誤りであるところのバーストエラーを、修整や補間の行い易い離散的な誤りであるところのランダムエラーに変換することである。

【００３４】

加えて、画像の画面内の粗密による情報量の発生の偏りを均一化する目的を重視する場合には上記圧縮処理の前に本処理工程を持ってくると、ランレングス等の可変長符号を用いた場合の都合が良い。

【００３５】

これを受けて、データ・シャフリングの復元のためのデータ識別（ID）情報を付加する。このID付加動作にて付加されたIDは、同時に記録しておいた上記システムのモード情報等と共に再生時の逆圧縮処理（情報量伸張処理）の際に補助情報として利用する。これらのデータの再生時の誤りを低減するためにエラー訂正(ECC)情報を付加する。

【００３６】

このような冗長信号の付加までを、映像と音声等の情報毎に対応する独立の記録エリア毎に処理する。上記のように、ID情報やECC 情報が付加された画像信号は、記録部58により、磁気テープ等の記録媒体に記録されるとともに、後述する第１のメモリ38に一時的に記憶される。

【００３７】

一方、ビデオ処理部34にて多重化された画像データは、D/A 変換器56によって、ディジタルーアナログ変換され、電子ビューファインダ54で操作者により観察される。また、操作者は第２の操作部52を介して、様々な操作情報をシステムコントローラ50に送信し、システムコントローラ50は、該操作情報によって、VCR全体を制御するようになっている。

【００３８】

また、ビデオ処理部34にて多重化された画像データは、第２のメモリ40に出力され、一時的に記憶される。前述した第１のメモリ38と、二のメモリ40とは、それぞれ、第１のメモリ制御回路46と、第２のメモリ制御回路48とを介し、システムコントローラ50により動作制御されている。

【００３９】

第１のデータセレクタ42は、前述した第１のメモリ38と、第２のメモリ40からのデータを選択して、第２の1394インターフェイス44に受け渡す、あるいは、第２の1394インターフェイス44からのデータを選択して、第１のメモリ38と、第２のメモリ40とのどちらかに受け渡す。

【００４０】

上記動作により、VCR28 における第２の1394インターフェイス44からは、圧縮

された画像データと非圧縮の画像データとが、操作者により選択されて出力できるようになっている。

【００４１】

第２の1394インターフェイス44は、1394シリアルバスを通じて、VCR28 を制御するためのコマンドデータを受信する。受信されたコマンドデータは、第１のデータセレクタ42を通じて、システムコントローラ50に入力される。システムコントローラ50は、上記のコマンドデータに対するレスポンスデータを作成して、第１のデータセレクタ42、及び、第２の1394インターフェイス44を通じ、1394シリアルバスに該データを送出する。

【００４２】

本実施の形態では、例えば、プリンタ60は、画像の印刷出力装置として動作する。第３の1394インターフェイス62は、1394シリアルバス上に転送される画像信号と、1394シリアルバスを通じて該プリンタ60を制御するためのコマンドデータとを受信する。また、第３の1394インターフェイス62は、該コマンドに対するレスポンスデータを送信する。

【００４３】

受信された画像データは、第２のデータセレクタ64を通じて、第２のデコーダ70に入力される。第２のデコーダ70は、該入力された画像データをデコードして、画像処理部74に出力する。画像処理部74は、デコードされた画像データを第３のメモリ72に一時的に記憶する。

【００４４】

一方、受信されたコマンドデータは、第２のデータセレクタ64を通じて、プリンタコントローラ68に入力される。プリンタコントローラ68は、該コマンドデータによりドライバ76による紙送り制御や、プリンタヘッド78の位置制御など、様々な印刷に関する制御を行なう。また、プリンタコントローラ68は、第３のメモリ72に一時的に記憶された画像データを、印刷データとして、プリンタヘッド78に送信し、印刷動作を行わせる。

【００４５】

上述したように、本実施の形態に係る、第１から第３の1394インターフェイス

14,44,62は、それぞれ、1394シリアルバスのノードを構成する。第１の1394インターフェイス14は、コントロールノード、または、コントローラとして動作し、第２の1394インターフェイス44は、画像データのソースノードとして動作し、第３の1394インターフェイス62は、デスティネーションノードとして動作する。

【００４６】

以下に、図２を用いて、本実施の形態のデータ転送動作の概要を説明する。図２において、200 はコントローラ、202 はソースノード、204 はデスティネーションノード、206 はソースノード内部のサブユニット、208 は画像データ等のobject、210 はデスティネーションノード内部の第１のメモリ空間、212 は第１のコネクション、214 はデスティネーションの第ｎのメモリ空間、216 は第ｎのコネクション、である。

【００４７】

コントローラ200 は、データ転送を行うソースノード202 と、１つ以上のデスティネーションノード204 との間の論理的な接続（以下、コネクション）を確立するためのコネクション識別データを設定するノードである。コントローラ200 は、ソースノード202 、及び、デスティネーションノード204 と独立したノードであってもよいし、ソースノード、または、デスティネーションノードとコントローラとが同じであってもかまわない。

【００４８】

後者の場合、コントローラと同じノードである、ソースノード、または、デスティネーションノードと、コントローラとの間のトランザクションは、不要である。本実施の形態では、コントローラ200 がソースノード202 、及び、デスティネーションノード204 とは別のノードに存在する場合の例を示す。

【００４９】

本実施の形態のデータ通信装置においては、複数のコネクションを確立することが可能である。コントローラはユーザが選択したデスティネーションノードとソースノードに同一のコネクション識別データをAsynchronousパケットを使用して設定する。

【００５０】

ソースノード202 は、内部のサブユニット206 から画像データ等のobject208を、例えば、第１のコネクション212 を通じて、デスティネーションノード内部の第１のメモリ空間210 にAsynchronousパケットを使用して書き込む。このときに使用するAsynchronousパケットは、例えば特定のノードIDを指定するデータを持たず、マルチキャスト用のIDを指定したパケットである。

【００５１】

このパケットにはコントローラによって設定されたコネクション識別データがデータヘッダ情報として書き込まれており、デスティネーションノードは第１のメモリ空間に書き込まれたデータのデータヘッダを解析し、コントローラによって設定されたコネクション識別情報を持つデータパケットである場合、データヘッダ情報を取り除いたデータを内部バッファに書き込んでいく。

【００５２】

デスティネーションノードが複数存在する場合も、例えば、特定のノードIDを指定しないマルチキャスト用のIDを使用していることにより、同時に複数のノードに対してデータ転送を行うことができる。

【００５３】

データの転送が終了すると、コントローラ200 はソースノード202 、デスティネーションノード204 に設定したコネクション識別データをクリアしてデータ転送を終了する。

【００５４】

次に、図３を用いて、Asynchronousパケットについて説明する。本実施の形態に係るAsynchronousパケットは、例えば、4 byte,(32 bits 、以下クアッドレットと称する）を単位とするデータパケットである。Asynchronousパケットは、送信先のノードIDを指定するパケットフォーマットと、Asynchronous Stream と呼ばれるチャンネル番号を指定するパケットフォーマットの２種類のフォーマットが存在する。本実施の形態では、オブジェクトデータを転送するパケットフォーマットとしていずれか一方のフォーマットを選択することができる。

【００５５】

図３(a) に示すパケットフォーマットは、ノードIDを指定するフォーマットで

ある。

最初の16 bits はdestination __IDフィールドであり、該フィールドは受信先のノードIDを示す。

次の6 bitsのフィールドは、トランザクション・ラベル(tl)フィールドであり、各トランザクション固有のタグである。

【００５６】

次の2 bitsのフィールドは、リトライ(rt)コードであり、パケットがリトライを試みるかどうかを指定する。

次の4 bitsのフィールドは、トランザクションコード(tcode) である。tcodeは、パケットのフォーマットや、実行しなければならないトランザクションのタイプを指定する。

【００５７】

本実施の形態においては、例えば、この値が0001_2 である、データブロックの書き込みリクエストのトランザクションを用いる。

次の4 bitsのフィールドは、プライオリティ(pri) フィールドであり、優先順位を指定する。本実施の形態においては、Asynchronousパケットを用いているので、このフィールドの値は0000_2 である。

【００５８】

次の16 bits はsource__IDフィールドであり、送信側のノードIDを示す。次の48 bits はdestination __offsetフィールドであり、パケットの受信先ノードアドレスの、下位48 bits がこのフィールドによって指定される。

【００５９】

次の16 bits はdata__lengthフィールドであり、後述するデータフィールドの長さを、バイト単位で示している。

次の16 bits はextended__tcode フィールドであり、本実施の形態に用いられるデータブロックの書き込みリクエストトランザクションにおいては、この値は0000_{16}である。

【００６０】

次の32 bits はheader__CRC フィールドであり、上述したdestination __IDフ

ィールドからextended_tcode フィールドまでを、パケットヘッダと称し、該ヘッダパケットのエラー検出に用いられる。

【００６１】

次のフィールドは可変長のデータフィールドであり、該データフィールドをパケットのペイロードと称する。本実施の形態においては、該データフィールドがクアッドレットの倍数でない場合、クアッドレットに満たないビットには0 が詰められる。

次の32 bits のフィールドはdata_CRC フィールドであり、上記のheader_CRC フィールドと同様に、該データフィールドのエラー検出に用いられる。

【００６２】

図３(b) に示すパケットフォーマットは、チャンネル番号を指定するAsynchronous stream フォーマットである。

最初の16 bits はdata_lengthフィールドであり、後述するデータフィールドの長さを、バイト単位で示している。

【００６３】

次の2 bitsはtag フィールドであり、この値は00_2 である。

次の6 bitsはchannel フィールドであり、このパケットのチャンネル番号を表す。受信ノードはこのチャンネル番号を用いてパケットを識別する。

次の4 bitsはトランザクションコード(tcode) である。Asynchronous stream パケットでは、A_{16} である。

【００６４】

次の4 bitsはSynchronization code(sy)フィールドであり、このパケットを使用するアプリケーションによって、この値は決定される。

次の32 bits はheader_CRC フィールドであり、上述したdata_lengthフィールドからsyフィールドフィールドまでを、パケットヘッダと称し、該ヘッダパケットのエラー検出に用いられる。

【００６５】

次のフィールドは可変長のデータフィールドであり、該データフィールドをパケットのペイロードと称する。本実施の形態においては、該データフィールドが

クアッドレットの倍数でない場合、クアッドレットに満たないビットには 0が詰められる。

【００６６】

次の32 bits のフィールドはdata__CRC フィールドであり、上記のheader__CRC フィールドと同様に、該データフィールドのエラー検出に用いられる。

次に、図４を用いて、上述した、コントローラ200 、ソースノード202 、デスティネーションノード204 の間で行われるAsynchronousトランザクションについて説明する。本実施の形態では１つのソースノードと１つのデスティネーションノードの間にコネクションを設定した場合を説明する。

【００６７】

コントローラはユーザーが選択したデスティネーションノードに対して、コネクションを設定するためのSET DESTINATION コマンドパケットを送信する。このパケットにはコントローラノードのROM 内に書き込まれているnode__vendor__id,chip __id__hi,chip __id__loのデータとこのコネクションに割り当てられたconnection__id が書き込まれている。

【００６８】

1394ノードが持つnode__vendor__id,chip __id__hi,chip __id__loを組み合わせた64ビットのデータはEUI-64(Extended Unique Identifier, 64bits)と呼ばれ、そのノード固有のデータであり、１つの通信システム内に同じEUI-64を持つ1394ノードは存在しない。本実施の形態ではEUI-64とconnection__idを組み合わせたデータをコネクション識別データとして使用する。

【００６９】

バス上に複数のコントローラが存在する場合でも各コントローラのEUI-64とconnection__idの組み合わせをコネクション識別データとすることにより、各コントローラは各コントローラがコネクションに割り当てたconnection__idのみを管理することでバス上で一意に定まるコネクション識別データを管理することができる。

【００７０】

図５に、上記コマンドパケットのフォーマット例を示す。ここに示すフォーマ

ットのデータは図３(a) のデータフィールドにAsynchronousパケットのペイロードとしてセットされてデスティネーションノードに送信される。

【００７１】

ctype フィールドはコマンドの種類を示す物であり、表１に示すコマンドタイプを指定する。SET DESTINATION コマンドの場合はControl を指定する。subunit _type,subunit ID フィールドは、指定したノード内のどのユニットに対してのコマンドなのかを示すフィールドである。opcode,operandフィールドは実際のコマンドを内容を指定するフィールドである。

【００７２】

【表１】

値	コマンドタイプ	意味
0	Control	制御コマンド
1	Status	機器の状態問い合わせ
2	Inquiry	該当コマンドのサポート状況問い合わせ
3	Notify	機器の状態変化の確認

【００７３】

図７に、上記SET DESTINATION コマンドパケットに使用されるopcode,operandの例を示す。opcodeフィールドには、デスティネーションに対してコネクション識別データを設定するためのパケットであることを示すコードがセットされる。

【００７４】

operand [0] ～operand [2] ,operand [3] ,operand [4] ～operand [7] のフィールドにはそれぞれコントローラノードのROM 内に書き込まれているnode_vendor_id,chip _id_hi,chip _id_loのデータがセットされる。operand [8] にはコントローラによって管理されるconnection_id 値がセットされる。コントローラからデスティネーションに対してこのパケットを送信する場合は残りのoperand のフィールドにはダミーデータをセットする。

【００７５】

上記SET DESTINATION コマンドパケットを受け取ったデスティネーションノー

ドは、コントローラノードにSET DESTINATION レスポンスパケットを送信する。

図６に、このレスポンスパケットのフォーマット例を示す。ここに示すフォーマットのデータは図３のデータフィールドにAsynchronousパケットのペイロードとしてセットされてコントローラノードに送信される。

【００７６】

responseフィールドは、レスポンスの種類を示す物であり、表２に示すレスポンスタイプを指定する。subunit __type,subunit ID フィールドは、ノード内のどのユニットからのレスポンスなのかを示すフィールドである。opcode,operandフィールドはレスポンスデータを指定するフィールドである。

【００７７】

【表２】

値	レスポンスタイプ	意味
8	Not Implemented	該当コマンドはサポートされていない
9	Accepted	コマンドを受け付けた
A_{16}	Rejected	コマンドを拒否した
F_{16}	Interim	後でレスポンスを返す

【００７８】

デスティネーションがデータトランザクションのコネクションを設定することが可能な場合は、responseフィールドにInterim を設定し（ソースノードとのデータトランザクションが終了した時点でAcceptedのレスポンスを再度コントローラノードに送信する）、不可能な場合はRejectedを設定する。opcode,operandフィールドは図７に示すフォーマットで、opcodeにはSET DESTINATION レスポンスパケットであることを示すコードを、node__vendor__id,chip __id__hi,chip __id__lo,connection __idフィールドはコントローラから指定されたデータを設定する。

【００７９】

コネクション設定が可能な場合はmax __rec フィールドにIEEE1394で規定され

るデスティネーションノードのmax __rec 値（Asyncronous の１パケットで受信できるデータサイズを示すデータである）を設定し、buffer__size フィールドにデータ受信に使用できるバッファサイズを設定する。

【００８０】

コネクション設定が不可能な場合、max __rec,buffer__size フィールドはコントローラからのコントロールコマンドパケットに設定されていたダミーデータと同じデータを設定する。status__infoフィールドにはコマンドの実行状況を設定する。表３にSET DESTINATION レスポンスパケットのstatus値の例を示す。コネクション設定可能な場合はSuccess を、不可能な場合はBusyを設定する。

【００８１】

【表３】

Value	meaning
00_{16}	Success
01_{16}	Aborted
11_{16}	Busy
12_{16}	Serial Bus Error
13_{16}	Hardware trouble

【００８２】

デスティネーションノードからInterim のSET DESTINATION レスポンスパケットを受信した後、コントローラノードはユーザが選択したソースノードに対して、コネクションを設定するためのSET SOURCEコマンドパケットを送信する。

【００８３】

このパケットには上記デスティネーションノードに設定したnode__vendor__id,chip __id__hi,chip __id__lo,connection __idが書き込まれている。このコマンドパケットのフォーマットも図５に示すものでctype フィールドにはControlが設定される。また上記フォーマットのデータは図３(a) のデータフィールドにAsynchronousパケットのペイロードとしてセットされてソースノードに送信される。

【００８４】

図８に上記SET SOURCEコマンドパケットに格納するデータの一例を示す。opcodeフィールドには、ソースノードに対してコネクション識別データを設定するためのパケットであることを示すコードがセットされる。

【００８５】

operand[0]～operand[2],operand[3],operand[4]～operand[7],operand[8] のフィールドには上記デスティネーションノードに設定したnode__vendor__id,chip __id__hi,chip __id__lo,connection __idのデータがセットされる。

【００８６】

operand[9]のmax __rec フィールド及びoperand[10] ～operand[12] のbuffer __size には、デスティネーションノードからのSET DESTINATION レスポンスパケットにセットされていたmax __rec,buffer__size のデータがセットされる。operand[13] のnumber__of__destinationsフィールドにはデスティネーションの数がセットされる。

【００８７】

図４に示すフローでは、１対１のトランザクションなので、この場合は１がセットされる。operand[14] のstatus__infoにはダミーデータがセットされる。

上記SET SOURCEコマンドパケットを受け取ったソースノードは、コントローラノードにSET SOURCEレスポンスパケットを送信する。SET SOURCEレスポンスパケットのフォーマットは図６に示すフォーマットである。また、上記フォーマットのデータは図３(a) のデータフィールドにAsynchronousパケットのペイロードとしてセットされてコントローラノードに送信される。

【００８８】

ソースノードがデータトランザクションのコネクションを設定することが可能な場合は、responseフィールドにAcceptedを設定し、不可能な場合はRejectedを設定する。opcode,operandフィールドは図８に示すデータが格納される。opcodeにはSET SOURCEレスポンスパケットであることを示すコードを、node__vendor__id,chip __id__hi,chip __id__lo,connection __idフィールドはコントローラから指定されたデータを設定する。

【００８９】

データトランザクションのコネクションを設定できる場合、パケットからコネクション識別データを取りだしてノード内部のバッファに記憶する。max __rec, buffer__sizeフィールドはSET SOURCEコマンドパケットにセットされていたデータを設定する。

【００９０】

本実施の形態では、ソースノードと複数のデスティネーションノード間でのコネクションを設定できるが、Asynchronous受信用のFIFO容量等の関係で、各ソースノードは同時に接続できるデスティネーションノードの数が制限される。

【００９１】

SET SOURCEコマンドパケットのnumber__of__destinationsフィールドの値を調べて、接続可能な数値以下の場合はその値をSET SOURCEレスポンスパケットのnumber__of__destinationsフィールドにセットする。接続可能な数値を超える場合は、接続可能なノード数の値をセットする。

【００９２】

status__infoフィールドはコマンドの実行状況を設定する。表４にSET SOURCEレスポンスパケットのstatus値の例を示す。コネクション設定可能な場合で、デスティネーションノード数が接続可能なノード数以下の場合はSuccess を、接続可能なノード数を超えている場合はToo many destinations を設定する。コネクション設定不可能な場合はBusyを設定する。

【００９３】

【表４】

Value	meaning
00_{16}	Success
02_{16}	Too many destinations
11_{16}	Busy
12_{16}	Serial Bus Error
13_{16}	Hardware trouble

【００９４】

ソースノードからAcceptedのSET SOURCEレスポンスパケットを受信した後、コントローラノードはSET __SOURCEレスポンスパケットのstatus__infoフィールドを調べてSuccess であることを確認すると、ソースノードに対してOBJECT SENDコマンドパケットを送信する。

【００９５】

ソースノードでは本実施の形態で示されない何らかの手段によって、あらかじめ送信を行うデータが選択されているものとする。このパケットには上記ソースノードに設定したnode__vendor__id,chip __id__hi,chip __id__lo,connection __idが書き込まれている。

【００９６】

このコマンドパケットのフォーマットも図５に示すものでctype フィールドにはControl が設定される。また、上記フォーマットのデータは図３(a) のデータフィールドにAsynchronousパケットのペイロードとしてセットされてソースノードに送信される。

【００９７】

図９に、上記OBJECT SEND コマンドパケットに格納されるデータの一例を示す。opcodeフィールドには、ソースノードに対して選択されているオブジェクトデータの送信開始を指示するパケットであることを示すコードがセットされる。

【００９８】

operand[0]～operand[2],operand[3],operand[4]～operand[7],operand[8] のフィールドには上記ソースノードに設定したnode__vendor__id,chip __id__hi,chip __id__lo,connection __idのデータ、すなわち、コントローラのEUI-64がセットされる。operand[9]のsubfunction フィールドにはこのコマンドパケットで指示される実際の動作を表すコードがセットされる。このコードは表５に示すようなコードで、データ送信の開始を指示する場合はsendがセットされる。operand[10] のstatus__infoにはダミーデータがセットされる。

【００９９】

【表５】

subfunction	Value	Action
send	00_{16}	Perform a normal operation
abort	02_{16}	Perform a “abort command” operation

【０１００】

上記OBJECT SEND コマンドパケットを受け取ったソースノードは、コントローラノードにOBJECT SEND レスポンスパケットを送信する。OBJECT SEND レスポンスパケットのフォーマットも図６に示すフォーマットである。また上記フォーマットのデータは図３(a) のデータフィールドにAsynchronousパケットのペイロードとしてセットされてコントローラノードに送信される。

【０１０１】

ソースノードはOBJECT SEND コマンドパケットに設定されているコネクション識別データを調べて、自ノードに設定されているデータと一致する場合でデータ送信開始が可能な場合はresponseフィールドにInterim を設定し（デスティネーションノードとのデータトランザクションが終了した時点でAcceptedのレスポンスを再度コントローラノードに送信する）、そうでない場合はRejectedを設定する。

【０１０２】

opcode,operandフィールドは、図９に示すデータが格納される。opcodeにはOBJECT SEND レスポンスパケットであることを示すコードを、node__vendor__id,chip __id__hi,chip __id__lo,connection __id,subfunctionフィールドにはコントローラからのOBJECT SEND コマンドパケットに設定されていたデータを設定する。

【０１０３】

status__infoフィールドはコマンドの実行状況を設定する。表６にOBJECT SEND レスポンスパケットのstatus値の例を示す。データ送信開始が可能な場合はSuccess を設定し、そうでない場合は送信開始が不可能な理由に応じたデータを設定する。

【０１０４】

【表６】

Value	meaning
00_{16}	Success
01_{16}	Aborted
11_{16}	Busy
12_{16}	Serial Bus Error
13_{16}	Hardware trouble
14_{16}	Unknown connection id

【０１０５】

以上のAsynchronousトランザクションをコントローラノード、デスティネーションノード、ソースノードの間で行うことにより、コネクションの設定が完了し、オブジェクトデータのトランザクションが開始される。上記SET DESTINATION, SET SOURCE,OBJECT SENDパケットはそれぞれ図３(a) のdestination _IDフィールドに受信ノードのノードIDを指定したパケットを使用して各ノードから送信が行われる。

【０１０６】

コントローラノードはconnection_id管理用のテーブルを持ち、コネクションに使用されているconnection_idを管理している。例えば、図１９に示すような、connection_idに対応したフラグレジスタ及びmax _rec,buffer_sizeを記憶するためのバッファを持ち、使用中のconnection_idのflagレジスタに使用中のビットをセットして、そのconnection_idに対するmax _rec 値及びbuffer_size値を保持する。このflagレジスタを調べて、使用されていないconnection_idを割り当てることにより、同時に複数のコネクションを設定することができる。

【０１０７】

次に、ソースノードとデスティネーション間で行われるオブジェクトデータのトランザクションについて説明する。

図１０に示すように、ソースノードでは送信オブジェクトとして128KB の静止画像データが選択されており、デスティネーションノードは32KBのデータ受信用のバッファを持つものとする。また、デスティネーションノードは512 バイトま

でのAsynchronousパケットを受信できるものとする。

【０１０８】

この場合、デスティネーションノードは上述のSET DESTINATION レスポンスでmax __rec フィールドに512 バイト、buffer__sizeフィールドに32KBを示すデータをセットしてコントローラにレスポンスを送信し、コントローラはこのデータをSET SOURCEコマンドを使用してソースノードに通知する。

【０１０９】

ソースノードは、選択されているオブジェクトをmax __rec 値を超えないsegment データに分割して（本実施の形態では256byte のsegment データ）各セグメントをデスティネーションに送信する。図１０では、オブジェクトデータを等しいサイズのsegment に分割している。

【０１１０】

図４のフローに示すように、セグメント単位でソースノードはオブジェクトデータを送信する。この送信にはAsynchronousトランザクションを使用するが、図３(a) のAsynchronousパケットを使用する場合、destination __idフィールドには特定のデスティネーションノードを示すノードIDではなく、マルチキャスト用のIDが設定される。

【０１１１】

本実施の形態では、例えば$FFDF_{16}$が設定され、オブジェクトデータの通信を行うために、destination __idがこの値にセットされているパケットをソースノードとデスティネーションノードとの間で送受信する。また、図３(b) のフォーマットのAsynchronous stream パケットフォーマットを使用する場合、特定のデスティネーションノードを示すノードIDではなく、所定のノードにより管理されたチャンネル番号がchannel フィールドにセットされる。

【０１１２】

この場合、コントローラがオブジェクトデータの転送前に、上記チャンネル番号をソースノードとデスティネーションに通知する。ソースノードとデスティネーションとは、このチャンネル番号がセットされたパケットを用いてオブジェクトデータの通信を行う。

【０１１３】

以下では、図３(a) のパケットフォーマットを使用した例を説明する。図１１に、Segment に分割されたオブジェクトデータの送信パケットフォーマット例を示す。このフォーマットのデータは図３(a) に示すdestination __idフィールドにマルチキャスト用のIDが設定されたAsynchronousパケットのData fieldにセットされてソースノードから送信される。

【０１１４】

node__vendor__id,chip __id__hi,chip __id__lo,connection __idフィールドには、コントローラから設定されたコネクション識別データが設定される。control __flags は、このデータパケットのタイプを示す情報がセットされる。表７にソースノードで設定するcontrol __flags の値の例を示す。

【０１１５】

【表７】

Value	meaning
00_{16}	Normal data
01_{16}	Buffer end
02_{16}	Object end

【０１１６】

sequence__numberフィールドには、このパケットで送信されるsegment データの通し番号がセットされる。Segmented object data の部分に、segment 分割されたオブジェクトのデータがセットされる。また、図３(b) に示すパケットを使用する場合も同様に、図１１に示すフォーマットがdata fieldにセットされる。

【０１１７】

まず、ソースノードの動作について説明する。

ソースノードは、送信したsegment データのトータルサイズをカウントし、コントローラからSET SOURCEコマンドで通知されたデスティネーションのバッファサイズに達するまで順次データの送信を行う。

【０１１８】

この場合の各パケットのcontrol __flags フィールドにはNormal data がセットされる。送信データのトータルサイズがデスティネーションのバッファサイズを超えない適当なサイズに達した場合、そのパケットのcontrol __flags フィールドにBuffer endをセットしてデータを送信する。

【０１１９】

図１０の例では、トータルサイズがデスティネーションのバッファサイズと等しくなるsequence number が127 のsegment の送信時に、control __flags をBuffer__end にセットする。その後、デスティネーションノードからレシーブレスポンスパケットが送信されるのを待機する。

【０１２０】

図１２にレシーブレスポンスのパケットフォーマット例を示す。このフォーマットのデータは、図３(a) に示すdestination __idフィールドにマルチキャスト用のIDが設定されたAsynchronousパケットのData fieldにセットされてデスティネーションノードから送信される。

【０１２１】

node__vendor__id,chip __id__hi,chip __id__lo,connection __idフィールドには、コントローラから設定されたコネクション識別データが設定される。control __flags は、このデータパケットのタイプを示す情報がセットされる。表８にデスティネーションノードで設定するcontrol __flags の値の例を示す。

【０１２２】

【表８】

Value	meaning
10_{16}	Receive success
11_{16}	Resend request

【０１２３】

control __flags の値がReceive success の場合は正しく受信できたデータのシーケンス番号が、Resend requestの場合は再送を希望するデータのシーケンス番号がセットされる。

【０１２４】

ソースノードは、destination __idにマルチキャスト用のIDが設定されているAsynchronousパケットを受信して、ペイロード中のコネクション識別データを調べて、自ノードに設定されているデータと一致する場合にcontrol __flags の値を調べる。この値がReceive success の場合、送信segment データのトータルカウントをクリアして次のsegment からデータ送信を開始する。

【０１２５】

図１０の例では、sequence number が255,383 のsegment データの送信時にcontrol __flags をBuffer endにセットして、同様にデスティネーションからのレスポンスを受け取った後にデータ送信を再開する。

【０１２６】

オブジェクトの最終データの送信時にはcontrol __flags をObject endに設定してソースノードはデータを送信する。Buffer endの場合と同様にソースノードはレシーブレスポンスを待機し、デスティネーションからのレシーブレスポンスのcontrol __flags がReceive success の場合、コントローラノードに対して、OBJECT SEND レスポンスパケットを送信する。

【０１２７】

このAsynchronousパケットのdestination __idにはコントローラノードのノードIDを設定して送信する。図６のresponseフィールドにはAcceptedをセットし、図９のopcode,operand[0] ～operand[9]にはOBJECT SEND コマンドを受け取った直後にコントローラに対して送信したInterim のOBJECT SEND レスポンスと同一のデータを設定する。

【０１２８】

operand[10] のstatus__infoフィールドにはデータ送信の終了状態を示すコードをセットする。正常に終了した場合はSuccess がセットされる。次に、デスティネーションノードの動作について説明する。

【０１２９】

デスティネーションノードは、destination __idフィールドにマルチキャスト用のIDが設定されているAsynchronousパケットを受信するとデータフィールド中

のコネクション識別データを調べて、自ノードに設定されているデータと一致した場合、受信用バッファの先頭アドレスから順次segment データを書き込んでいく。また、このときsequence number フィールドを調べて、欠落データの検出を行うことができる。

【０１３０】

データパケットのcontrol __flags を調べて、この値がNormal data の場合は次のデータパケットが送信されるのを待つ。control __flags の値がBuffer end の場合、受信用バッファに書き込んだデータを別のバッファ（ハードディスク等）にコピーして、バッファをクリアした後に、図１２に示すフォーマットのレシーブレスポンスパケットをマルチキャスト用IDを使用して送信する。

【０１３１】

このとき、control __flags にはReceive success を、sequence number にはBuffer endが設定されたデータパケットのsequence number をセットする。図１０の例では127 がセットされる。

【０１３２】

レシーブレスポンスを送信した後、デスティネーションノードはソースノードからデータの送信が再開されるのを待機し、データを受信すると再び受信用バッファの先頭から順次segment データを書き込んでいく。図１０の例の場合、sequence number が255,383 のsegment データを受信すると、同様にレシーブレスポンスを送信する。

【０１３３】

control __flags の値がObject endのsegment データを受信すると、デスティネーションノードは同様にレシーブレスポンスを送信する。Object endに対するレシーブレスポンスをマルチキャスト用IDを使用して送信した後、デスティネーションノードはコントローラノードに対してSET DESTINATION レスポンスパケットを送信する。

【０１３４】

このAsynchronousパケットのdestination __idにはコントローラノードのノードIDを設定して送信する。図６のresponseフィールドにはAcceptedをセットし、

図７のopcode,operand[0] ～operand[12] にはSET DESTINATION コマンドを受け取った直後にコントローラに対して送信したInterim のSET DESTINATION レスポンスと同一のデータを設定する。

【０１３５】

operand[13] のstatus__info フィールドにはデータ送信の終了状態を示すコードをセットする。正常に終了した場合はSuccess がセットされる。上述の手順でソースノードとデスティネーションノードの間のオブジェクト転送が完了する。

【０１３６】

ソースノードとデスティネーションノードからそれぞれOBJECT SEND,SET DESTINATION のレスポンスパケットを受信した後、コントローラノードはソースノード、デスティネーションノードに対してコネクションを解除するためのCLEAR CONNECTIONコマンドパケットを送信する。

【０１３７】

このコマンドパケットのフォーマットは図５に示すものでctype フィールドにはControl が設定される。また上記フォーマットのデータは図３(a) のデータフィールドにAsynchronousパケットのペイロードとしてセットされてソースノード、デスティネーションノードに送信される。

【０１３８】

図１３に、上記CLEAR CONNECTIONコマンドパケットのopcode,operandフィールドに格納するデータの一例を示す。opcodeフィールドには、コネクションの解除パケットであることを示すコードがセットされる。operand[0]～operand[2],operand[3],operand[4]～operand[7],operand[8] のフィールドには解除を行うコネクション識別データに使用されているnode__vendor__id,chip __id__hi,chip __id__lo,connection __idのデータがセットされる。

【０１３９】

operand[9]のstatus__infoにはコネクション解除の理由を示すコードがセットされる。表９にデスティネーションノードに対してCLEAR CONNECTIONコマンドを送信する場合のstatus__infoの値の例を示す。

【０１４０】

【表９】

Value	meaning
00_{16}	Transfer is over
01_{16}	Transfer is aborted
12_{16}	Serial Bus Error
20_{16}	Source busy
21_{16}	Too many Destinations
22_{16}	Source error
FF_{16}	No information

【０１４１】

また、表１０にソースノードに対してCLEAR CONNECTIONコマンドを送信する場合のstatus_infoの値の例を示す。

【０１４２】

【表１０】

Value	meaning
00_{16}	Transfer is over
01_{16}	Transfer is aborted
12_{16}	Serial Bus Error
20_{16}	Destination busy
22_{16}	Destination error
FF_{16}	No information

【０１４３】

正常にデータ転送が終了している場合はソースノード、デスティネーションノードのどちらに対してもstatus_infoフィールドにはTransfer is overがセットされる。

【０１４４】

上記CLEAR CONNECTIONコマンドを受信したソースノード及びデスティネーショ

ンノードはコネクション識別データがそれぞれ自ノードに設定されているデータと一致する場合、それぞれのノードが内部バッファに記憶していたコネクション識別データをクリアして、コントローラノードに対してCLEAR CONNECTIONレスポンスパケットを送信する。

【０１４５】

このレスポンスパケットのフォーマットは図６に示すもので正常にコネクションの解除が行われた場合はresponseフィールドにはAcceptedが設定される。また上記フォーマットのデータは図３(a) のデータフィールドにAsynchronousパケットのペイロードとしてセットされてコントローラノードに送信される。

【０１４６】

上記CLEAR CONNECTIONレスポンスパケットのopcode,operandフィールドに格納されるデータの一例を図１３に示す。opcodeにはCLEAR CONNECTIONレスポンスパケットであることを示すコードを、node__vendor__id,chip __id__hi,chip __id__lo,connection __id,subfunctionフィールドには解除したコネクション識別データに使用していたnode__vendor__id,chip __id__hi,chip __id__lo,connection __idを設定する。

【０１４７】

status__infoフィールドはコマンドの実行状況を設定する。表１１にCLEAR CONNECTIONレスポンスパケットのstatus値の例を示す。正常にコネクションを解除した場合はSuccess をセットする。

【０１４８】

【表１１】

Return value	meaning
00_{16}	Success
11_{16}	Busy
12_{16}	Serial Bus Error
14_{16}	Unknown connection id

【０１４９】

ソースノード、デスティネーションノードからAcceptedのCLEAR CONNECTIONレスポンスパケットを受信したコントローラノードは、コネクションを解除したconnection__idフラグレジスタの使用中のビットをクリアする。以上の手順で、ソースノードとデスティネーションノードの間に設定された論理的なコネクションが解除され、全てのトランザクションが終了する。

【０１５０】

このように、コントローラが同一のコネクション識別データをデスティネーションノード及びソースノードに設定することによって、ソースノードとデスティネーションノード間に論理的なコネクションを設定することができ、データ転送処理はコントローラーが介在しない、ソース・デスティネーション間のみのトランザクションで行うことができる。

【０１５１】

（第２の実施の形態）

次に、図１４を用いて第２の実施の形態を説明する。本実施の形態では図２に示したコントローラ200 、ソースノード202 、デスティネーションノード204 との間でコネクションを設定してデータ転送を行う。

【０１５２】

コントローラはユーザーが選択したデスティネーションノードに対して、コネクションを設定するためのSET DESTINATION コマンドパケットを送信する。このパケットにはコントローラノードのROM 内に書き込まれているnode__vendor__id,chip __id__hi,chip __id__loのデータとこのコネクションに割り当てられたconnection__idが書き込まれている。

【０１５３】

上記SET DESTINATION コマンドパケットのフォーマットは図５に示すものでctype フィールドにはControl が設定される。また上記フォーマットのデータは図３(a) のデータフィールドにAsynchronousパケットのペイロードとしてセットされてデスティネーションノードに送信される。

【０１５４】

図１５に、上記SET DESTINATION コマンドパケットに使用されるopcode,opera

ndの例を示す。opcodeフィールドには、デスティネーションに対してコネクション識別データを設定するためのパケットであることを示すコードがセットされる。

【０１５５】

operand[0]～operand[2],operand[3],operand[4]～operand[7]のフィールドにはそれぞれコントローラノードのROM 内に書き込まれているnode__vendor__id,chip __id__hi,chip __id__loのデータがセットされる。

【０１５６】

operand[8]にはコントローラによって管理されるconnection__id値がセットされる。コントローラからデスティネーションに対してこのパケットを送信する場合は残りのoperand のフィールドにはダミーデータをセットする。

【０１５７】

上記SET DESTINATION コマンドパケットを受け取ったデスティネーションノードは、コントローラノードにSET DESTINATION レスポンスパケットを送信する。上記SET DESTINATION レスポンスパケットのフォーマットは図６に示すもので、図３(a) のデータフィールドにAsynchronousパケットのペイロードとしてセットされてコントローラノードに送信される。

【０１５８】

デスティネーションがデータトランザクションのコネクションを設定することが可能な場合はresponseフィールドにInterim を設定し（ソースノードとのデータトランザクションが終了した時点でAcceptedのレスポンスを再度コントローラノードに送信する）、不可能な場合はRejectedを設定する。

【０１５９】

opcode,operandフィールドは図１５に示すフォーマットで、opcodeにはSET DESTINATION レスポンスパケットであることを示すコードを、node__vendor__id,chip __id__hi,chip __id__lo,connection __idフィールドはコントローラから指定されたデータを設定する。destination __node__vendor__id,destination__chip__id__hi,destination__chip__id__loフィールドにはデスティネーションノードのROM に書き込まれているnode__vendor__id,chip __id__hi,chip __id__loを

設定する。

【０１６０】

コネクション設定が可能な場合はmax __rec フィールドにIEEE1394で規定されるデスティネーションノードのmax __rec 値（Asyncronous の１パケットで受信できるデータサイズを示すデータである）を設定し、buffer__size フィールドにデータ受信に使用できるバッファサイズを設定する。

【０１６１】

コネクション設定が不可能な場合、max __rec,buffer__size フィールドはコントローラからのコントロールコマンドパケットに設定されていたダミーデータと同じデータを設定する。status__infoフィールドにはコマンドの実行状況を設定する。SET DESTINATION レスポンスパケットのstatus値の例は表３に示した通りである。コネクション設定可能な場合はSuccess を、不可能な場合はBusyを設定する。

【０１６２】

デスティネーションノードからInterim のSET DESTINATION レスポンスパケットを受信した後、コントローラノードはユーザが選択したソースノードに対して、コネクションを設定するためのSET SOURCEコマンドパケットを送信する。このパケットには上記デスティネーションノードに設定したnode__vendor__id,chip __id__hi,chip __id__lo,connection __idが書き込まれている。

【０１６３】

このコマンドパケットのフォーマットは図５に示すもので、ctype フィールドにはControl が設定される。また上記フォーマットのデータは図３(a) のデータフィールドにAsynchronousパケットのペイロードとしてセットされてソースノードに送信される。

【０１６４】

図１６に上記SET SOURCEコマンドパケットの一例を示す。opcodeフィールドには、ソースノードに対してコネクション識別データを設定するためのパケットであることを示すコードがセットされる。

【０１６５】

node__vendor__id,chip __id__hi,chip __id__lo,connection __idフィールドには上記デスティネーションノードに設定したnode__vendor__id,chip __id__hi,chip __id__lo,connection __idのデータがセットされる。max __rec フィールド及びbuffer__size フィールドには、デスティネーションノードからのSET DESTINATION レスポンスパケットにセットされていたmax __rec,buffer__sizeのデータがセットされる。number__of__destinationsフィールドにはデスティネーションの数がセットされる。

【０１６６】

図１４に示すフローでは１対１のトランザクションなので、この場合は１がセットされる。source__node__vendor__id,source __chip__id__hi,source __chip__id__lo,status __infoにはダミーデータがセットされる。

【０１６７】

上記SET SOURCEコマンドパケットを受け取ったソースノードは、コントローラノードにSET SOURCEレスポンスパケットを送信する。SET SOURCEレスポンスパケットのフォーマットは図６に示すフォーマットである。また上記フォーマットのデータは図３(a) のデータフィールドにAsynchronousパケットのペイロードとしてセットされてコントローラノードに送信される。

【０１６８】

ソースノードがデータトランザクションのコネクションを設定することが可能な場合はresponseフィールドにAcceptedを設定し、不可能な場合はRejectedを設定する。opcode,operandフィールドは図１６に示すフォーマットで、opcodeにはSET SOURCEレスポンスパケットであることを示すコードを、node__vendor__id,chip __id__hi,chip __id__lo,connection __idフィールドはコントローラから指定されたデータを設定する。

【０１６９】

データトランザクションのコネクションを設定できる場合、パケットからコネクション識別データを取りだしてノード内部のバッファに記憶する。source__node__vendor__id,source __chip__id__hi,source __chip__id__loフィールドにはソースノードのROM に書き込まれているnode__vendor__id,chip __id__hi,chip

__id__loを設定する。

【０１７０】

max __rec,buffer__sizeフィールドはSET SOURCEコマンドパケットにセットされていたデータを設定する。本実施の形態ではソースノードと複数のデスティネーションノード間でのコネクションを設定できるが、Asynchronous受信用のFIFO容量等の関係で、各ソースノードは同時に接続できるデスティネーションノードの数が制限される。

【０１７１】

SET SOURCEコマンドパケットのnumber__of__destinationsフィールドの値を調べて、接続可能な数値以下の場合はその値をSET SOURCEレスポンスパケットのnumber__of__destinationsフィールドにセットする。接続可能な数値を超える場合は、接続可能なノード数の値をセットする。status__infoフィールドはコマンドの実行状況を設定する。

【０１７２】

SET SOURCEレスポンスパケットのstatus値の例は表４に示す通りである。コネクション設定可能な場合で、デスティネーションノード数が接続可能なノード数以下の場合はSuccess を、接続可能なノード数を超えている場合はToo many destinations を設定する。コネクション設定不可能な場合はBusyを設定する。

【０１７３】

ソースノードからAcceptedのSET SOURCEレスポンスパケットを受信した後、コントローラノードはSET __SOURCEレスポンスパケットのstatus__infoフィールドを調べてSuccess であることを確認すると、ソースノードに対してOBJECT SENDコマンドパケットを送信する。OBJECT SEND コマンドの動作については第１の実施の形態と同様であるので説明を省略する。

【０１７４】

以上のAsynchronousトランザクションをコントローラノード、デスティネーションノード、ソースノードの間で行うことにより、コネクションの設定が完了し、オブジェクトデータのトランザクションが開始される。上記SET DESTINATION, SET SOURCE,OBJECT SENDパケットはそれぞれ図３(a) のdestination __IDフィー

ルドに受信ノードのノードIDを指定したパケットを使用して各ノードから送信が行われる。

【０１７５】

コントローラノードはconnection__id管理用のテーブルを持ち、コネクションに使用されているconnection__idを管理している。たとえば図２０に示すような、connection__idに対応したflagレジスタ及びmax __rec,buffer__size，デスティネーションノード数，デスティネーションノード・ソースノードのEUI-64の値を記憶するためのデータバッファを持ち、使用中のconnection__idのflagレジスタに使用中のビットをセットして、そのconnection__idに対する上述の各データを保持する。このフラグレジスタを調べて、使用されていないconnection__idを割り当てることにより、同時に複数のコネクションを設定することができる。

【０１７６】

OBJECT SEND コマンドに従って、第１の実施の形態と同様にオブジェクトデータの転送が開始される。図１４中のData Segment __m+1 のデータをデスティネーションノードが受信した後にバスリセットが発生した場合、バス上の各ノードはIEEE1394の規格に定められた手順でバスの再構築を行う。これにより、ソースノードとデスティネーションノードのノードIDは再設定される。

【０１７７】

バスの再構築が終了すると、デスティネーションノードはマルチキャストIDをdestination __idフィールドに設定したAsynchronousパケット（図３(a) に示す）を使用して、図１２に示すフォーマットのレシーブレスポンスパケットを送信する。また、オブジェクトデータのトランザクションを図３(b) に示すパケットフォーマットにて行っている場合、デスティネーションは、所定のチャンネル番号をchannel フィールドに格納したパケットを用いてレシーブレスポンスパケットを送信する。

【０１７８】

node__vendor__id,chip __id__hi,chip __id__lo,connection __idフィールドには、それぞれバスリセット前にコントローラノードによって設定されたnode__vendor__id,chip __id__hi,chip __id__lo,connection __idの値をセットする。

control __flags にはResend requestを、sequence number にはバスリセット前に受信できたsegment のシーケンス番号に１を加えたm+2 をセットする。

【０１７９】

バスリセット後、ソースノードはデスティネーションノードからResend requestが設定されたレシーブレスポンスパケットが送信されるのを待機する。受信したレシーブレスポンスパケットのnode__vendor__id,chip __id__hi,chip __id__lo,connection __idを調べて、バスリセット前に使用していたデータと一致した場合、要求されたsequense__numberのsegment データからデータの送信を再開する。

【０１８０】

バスリセット後、コントローラノードはソースノードとデスティネーションノードIDを管理するために、connection__idの管理用テーブルと、connection__idに対応したデータバッファからコネクションが設定されているデスティネーションノードとソースノードのEUI-64の値を読み出して、バスリセットによって変更されたデスティネーションノードとソースノードのノードIDを検出する。

【０１８１】

このノードIDの検出は、例えばバス上の全てのノードに対して、そのノードのROM に書き込まれているEUI-64値をAsynchronousReadトランザクションを使用して読み出すことによって行うことができる。

【０１８２】

各ノードIDの検出が終了すると、コントローラノードはコネクションの再設定のためデスティネーションノードに対してSET DESTINATION コマンドパケットを、ソースノードに対してOBJECT SEND コマンドパケットを送信する。各コマンドパケットを受信したデスティネーションノード及び、ソースノードはそれぞれ対応するレスポンズパケットをコントローラに対して送信する。このときのresponseフィールドはInterim が設定される。

【０１８３】

上記の手順により、バスリセット後にデータ転送及び、コネクションの再設定が行われ、以後のデータ転送及びコネクションの解除は第１の実施の形態と同様

の手順で行うことができる。

【０１８４】

本実施の形態では、SET DESTINATION,SET SOURCEコマンドによってデスティネーションノード及びソースノードのEUI-64の値を取得しているが、SET DESTINATION,SET SOURCEコマンドは第１の実施の形態と同様のフォーマットのものを用いて、別のAsynchronousReadトランザクション等を使用してEUI-64の値を取得することも可能である。

【０１８５】

このように、コントローラが同一のバスリセットによって変化しないコネクション識別データをデスティネーションノード及びソースノードに設定することによって、ソースノードとデスティネーションノード間に論理的なコネクションを設定することができ、バスリセットが発生した場合も迅速にデータ転送の再開を行うことができる。

【０１８６】

（第３の実施の形態）

次に、図１７に示すように、ソースノードと複数のデスティネーションとの間にコネクションを設定してデータ転送を行う場合について説明する。

各デスティネーションは異なるサイズの受信バッファを持っている。デスティネーション__1 は32KB、デスティネーション__2 は48KB、デスティネーション__3 は64KBの受信バッファを持つ。また、それぞれのmax __rec 値は、デスティネーション__1 が512byte,デスティネーション__2 が1024byte, デスティネーション__3 が1024byteを示している。

【０１８７】

図１８を用いて、本実施の形態のデータ転送の手順について説明する。

コントローラはユーザーが選択したデスティネーションノードに対して順次、コネクションを設定するためのSET DESTINATION コマンドパケットを送信する。このコマンドパケットは第１の実施の形態に示したパケットと同様のフォーマットである。各デスティネーションノードに対して送信されるSET DESTINATION コマンドパケットのnode__vendor__id,chip __id__hi,chip __id__lo,connection

__idフィールドは全て同じ値がセットされている。

【０１８８】

SET DESTINATION コマンドパケットを受信した各デスティネーションノードはパケットからコネクション識別データを自ノード内のバッファにセットして、コントローラノードに対して第１の実施の形態と同様にSET DESTINATION レスポンスパケットを送信する。

【０１８９】

このとき、デスティネーション__1 は図７中のmax __rec フィールドに512byte を示すデータを、buffer__sizeフィールドに32KBを示すデータをセットして、Interim のレスポンスパケットを送信する。

【０１９０】

同様に、デスティネーション__2 はmax __rec フィールドに1024byteを示すデータを、buffer__sizeフィールドに48KBを示すデータを、デスティネーション__3 はmax __rec フィールドに1024byteを示すデータを、buffer__sizeフィールドに64KBを示すデータをセットして、Interim のレスポンスパケットを送信する。

【０１９１】

各デスティネーションノードからInterim のSET DESTINATION レスポンスパケットを受信した後、コントローラノードはユーザが選択したソースノードに対して、コネクションを設定するためのSET SOURCEコマンドパケットを送信する。

【０１９２】

このSET SOURCEコマンドパケットは、第１の実施の形態に示したパケットと同様のフォーマットである。このパケットには上記各デスティネーションノードに設定したnode__vendor__id,chip __id__hi,chip __id__lo,connection __idが書き込まれている。max __rec フィールドには、各ディスティネーションから受信したmax __rec 値の中で最小のデータをセットする。

【０１９３】

本実施の形態の場合、３つのデスティネーションから受信したmax __rec が示す値はそれぞれ512byte,1024byte,1024byte であるので、512byte を示すデータをセットする。また、buffer__sizeフィールドにも各ディスティネーションから

受信したbuffer__size値の中で最小のデータをセットする。本実施の形態の場合、３つのデスティネーションから受信したbuffer__sizeが示す値はそれぞれ32KB,48KB,64KBであるので、32KBを示すデータをセットする。number__of__destinationsフィールドにはデスティネーションの数を示す３がセットされる。

【０１９４】

上記SET SOURCEコマンドパケットを受け取ったソースノードは、コントローラノードにSET SOURCEレスポンスパケットを送信する。SET SOURCEレスポンスパケットは第１の実施の形態に示したパケットと同様のフォーマットであり、第１の実施の形態と同様に各フィールドにデータを設定してレスポンスパケットを送信する。

【０１９５】

ソースノードからAcceptedのSET SOURCEレスポンスパケットを受信した後、コントローラノードはSET __SOURCEレスポンスパケットのstatus__infoフィールドを調べてSuccess であることを確認すると、ソースノードに対してOBJECT SENDコマンドパケットを送信する。OBJECT SEND コマンドの動作については第１の実施の形態と同様であるので説明を省略する。

【０１９６】

以上のAsynchronousトランザクションをコントローラノード、各デスティネーションノード、ソースノードの間で行うことにより、ソースノードと複数のデスティネーションノード間のコネクションの設定が完了し、オブジェクトデータのトランザクションが開始される。上記SET DESTINATION,SET SOURCE,OBJECT SENDパケットはそれぞれ図３(a) のdestination __IDフィールドに受信ノードのノードIDを指定したパケットを使用して各ノードから送信が行われる。

【０１９７】

OBJECT SEND コマンドに従って、第１の実施の形態と同様にオブジェクトデータの転送が開始される。第１の実施の形態と同様にソースノードはdestinetion __idフィールドにマルチキャスト用のIDを設定したコネクション識別データを含むsegment データパケットをAsynchronous送信する。各デスティネーションノードは受信したマルチキャスト用IDを持つパケットのコネクション識別データを判

別し、自ノードに設定されているコネクション識別データと一致する場合segment データを内部の受信バッファに順次書き込んでいく。

【０１９８】

ソースノードは第１の実施の形態と同様に、送信したsegment データのトータルサイズをカウントし、コントローラからSET SOURCEコマンドで通知されたデスティネーションのバッファサイズに達するまで順次データの送信を行う。

【０１９９】

この場合の各パケットのcontrol __flags フィールドにはNormal data がセットされる。送信データのトータルサイズがデスティネーションのバッファサイズを超えない適当なサイズに達した場合、そのパケットのcontrol __flags フィールドにBuffer endをセットしてデータを送信する。

【０２００】

図１７の例では、トータルサイズがSET SOURCEコマンドパケットで通知されたデスティネーションのバッファサイズと等しくなるsequence number が127 のsegment の送信時に、control __flags をBuffer__end にセットする。その後、各デスティネーションノードからレシーブレスポンスパケットが送信されるのを待機する。

【０２０１】

各デスティネーションノードはデータパケットを受信するとパケット中のcontrol __flags を調べて、この値がNormal data の場合は次のデータパケットが送信されるのを待つ。control __flags の値がBuffer endの場合、受信用バッファに書き込んだデータを別のバッファ（ハードディスク等）にコピーして、バッファをクリアした後に、図１２に示すフォーマットのレシーブレスポンスパケットをマルチキャスト用IDを使用して送信する。このとき、control __flags にはReceive success を、sequence number にはBuffer endが設定されたデータパケットのsequence number をセットする。図１７の例では127 がセットされる。

【０２０２】

control __flags にBuffer endをセットしたsegment データを送信した後、ソースノードは各デスティネーションノードからレシーブレスポンスパケットを受

信して第１の実施の形態と同様に、続きのsegment データの転送を再開する。オブジェクトの最終segment データの送信時には、第１の実施の形態と同様にcontrol _flags フィールドにObject endをセットして同様にデスティネーションからのレシーブレスポンスを待機する。

【０２０３】

各デスティネーションはObject endがセットされたsegment データを受信すると、同様にレシーブレスポンスを送信する。Object endに対するレシーブレスポンスをマルチキャスト用IDを使用して送信した後、各デスティネーションノードはコントローラノードに対して第１の実施の形態と同様にSET DESTINATION レスポンスパケットを送信する。

【０２０４】

また、各デスティネーションからObject endのsegment データに対するレシーブレスポンスパケットを受信した後、ソースノードは第１の実施の形態と同様にコントローラノードに対してOBJECT SEND レスポンスパケットを送信する。上述の手順でソースノードと複数のデスティネーションノードの間のオブジェクト転送が完了する。

【０２０５】

ソースノードと各デスティネーションノードからそれぞれOBJECT SEND,SET DESTINATION のレスポンスパケットを受信した後、第１の実施の形態と同様にコントローラノードは各デスティネーションノードとソースノードに対してコネクションを解除するためのCLEAR CONNECTIONコマンドパケットを送信する。このCLEAR CONNECTIONコマンドパケットのは第１の実施の形態で示したパケットと同様のフォーマットである。

【０２０６】

CLEAR CONNECTIONコマンドパケットを受信した各デスティネーションノードとソースノードは第１の実施の形態と同様の手順でコネクションの解除を行って、CLEAR CONNECTIONレスポンスパケットをコントローラノードにそれぞれ送信する。

【０２０７】

各デスティネーションノード、ソースノードからAcceptedのCLEAR CONNECTIONレスポンスパケットを受信したコントローラノードは、コネクションを解除したconnection__idフラグレジスタの使用中のビットをクリアする。以上の手順で、ソースノードと複数のデスティネーションノードの間に設定された論理的なコネクションが解除され、全てのトランザクションが終了する。

【０２０８】

なお、第３の実施の形態では、第１の実施の形態と同様に、図３(b) に示すパケットフォーマットを用いてオブジェクトデータのトランザクションを行うこともとできる。

【０２０９】

また、第３の実施の形態において、オブジェクトデータの転送中にバスリセットが発生しても、各デスティネーションノードが第２の実施の形態と同様の処理を行うことによって、オブジェクトデータの転送を再開させることもできる。この場合、ソースノードは各デスティネーションノードのレシーブレスポンスパケットを受信し、最小となるシーケンス番号を加えた番号のセグメントデータから転送を再開する。また、コントローラは、ソースノードと全てのデスティネーションノードのノードIDを問い合わせて管理する。

【０２１０】

このように、コントローラが同一のコネクション識別データを複数のデスティネーションノード及びソースノードに設定することによって、ソースノードと複数のデスティネーションノード間に論理的なコネクションを設定することができ、データ転送処理はコントローラーが介在しない、ソース・デスティネーション間のみのトランザクションで行うことができる。

【０２１１】

また、各デスティネーションの持つ受信能力が異なる場合でも、その中の最も低い能力にあわせてデータ転送を行うことにより、複数のデスティネーションに対してデータ転送を行う場合も、煩雑な処理を必要としないデータ転送を行うことができる。

【０２１２】

（本実施の形態の他の実施形態）

本実施の形態は複数の機器（例えば、ホストコンピュータ、インタフェース機器、リーダ、プリンタ等）から構成されるシステムに適用しても１つの機器からなる装置に適用しても良い。

【０２１３】

また、上述した実施形態の機能を実現するように各種のデバイスを動作させるように、上記各種デバイスと接続された装置あるいはシステム内のコンピュータに対し、上記実施形態の機能を実現するためのソフトウェアのプログラムコードを供給し、そのシステムあるいは装置のコンピュータ（ＣＰＵあるいはＭＰＵ）に格納されたプログラムに従って上記各種デバイスを動作させることによって実施したものも、本実施の形態の範疇に含まれる。

【０２１４】

また、この場合、上記ソフトウェアのプログラムコード自体が上述した実施形態の機能を実現することになり、そのプログラムコード自体、およびそのプログラムコードをコンピュータに供給するための手段、例えばかかるプログラムコードを格納した記憶媒体は本実施の形態を構成する。かかるプログラムコードを記憶する記憶媒体としては、例えばフロッピーディスク、ハードディスク、光ディスク、光磁気ディスク、ＣＤ－ＲＯＭ、磁気テープ、不揮発性のメモリカード、ＲＯＭ等を用いることができる。

【０２１５】

また、コンピュータが供給されたプログラムコードを実行することにより、上述の実施形態の機能が実現されるだけでなく、そのプログラムコードがコンピュータにおいて稼働しているＯＳ（オペレーティングシステム）あるいは他のアプリケーションソフト等の共同して上述の実施形態の機能が実現される場合にもかかるプログラムコードは本実施の形態の実施形態に含まれることは言うまでもない。

【０２１６】

さらに、供給されたプログラムコードがコンピュータの機能拡張ボードやコンピュータに接続された機能拡張ユニットに備わるメモリに格納された後、そのプ

ログラムコードの指示に基づいてその機能拡張ボードや機能拡張ユニットに備わるＣＰＵ等が実際の処理の一部または全部を行い、その処理によって上述した実施形態の機能が実現される場合にも本実施の形態に含まれることは言うまでもない。

【０２１７】

【発明の効果】

上記説明したように、本発明においては、従来の通信方式による不便利性を解決することができる効果が得られる。また、リアルタイム性を必要としないデータの転送においても、高速にデータを転送することを簡便に可能とする効果が得られる。

また、本発明の他の特徴によれば、通信帯域をあまり使用しない場合に、多数の通信を同時に行なうことができる効果が得られる。

また、本発明の他の特徴によれば、データ転送中断により失われたデータを容易に検出することが可能であるとともに、該データ転送の中断からの復帰を、確実に、かつ、簡単に行なうことができる効果が得られる。

また、本発明によれば、複数のコントロール間でコネクション識別データが重複しないように調整する必要がないので、コントローラは、簡単に確実にコネクションを設定できる効果が得られる。

また、本発明の他の特徴によれば、複数のコントロールノードが個別に複数の論理的コネクションをソース、デスティネーション間に設定した場合も、個々のノードは、コネクションを設定したコントローラを上記ノード固有の情報であるEUI-64などの固有のノードＩＤにて判別することが可能となるので、個々のノードは、確実に論理的コネクションを識別できる効果が得られる。

本発明の他の特徴によれば、１個のコネクション識別データにより、容易に複数のデスティネーションに対してデーターを単一セグメントパケットで送信することができるため、バス上のトラフィックを低減する効果が得られる。

また、本発明のの他の特徴によれば、それぞれのデスティネーションの受信バッファが異なっていても、ソースは、それぞれのコントローラから指示された単一のバッファサイズのみを管理してデータ送信を行うことができるため、実装が

容易となる効果が得られる。

【図面の簡単な説明】

【図１】

第１の実施の形態の実施の形態を表すブロック図である。

【図２】

第１の実施の形態に係るデータ転送動作の概要を説明するための図である。

【図３】

Asynchronous トランザクションに使用されるパケットを示す図である。

【図４】

第１の実施の形態に係る各ノード間のコマンドやデータの授受のダイアグラムを示す図である。

【図５】

第１の実施の形態のコマンドパケットに用いられるデータフォーマットを示す図である。

【図６】

第１の実施の形態のコマンドパケットに対するレスポンスパケットに用いられるデータフォーマットを示す図である。

【図７】

第１の実施の形態においてデスティネーションに対して設定を行うコマンドパケットに用いられるデータを示す図である。

【図８】

第１の実施の形態においてソースに対して設定を行うコマンドパケットに用いられるデータを示す図である。

【図９】

第１の実施の形態のソースに対してデータ送信の指示を行うコマンドパケットに用いられるデータを示す図である。

【図１０】

第１の実施の形態のオブジェクトデータの転送を説明するための図である。

【図１１】

第１の実施の形態のオブジェクトデータの送信パケットのフォーマットを示す図である。

【図１２】

第１の実施の形態のオブジェクトデータの受信確認パケットのフォーマットを示す図である。

【図１３】

第１の実施の形態のコネクション解除を行うコマンドパケットに用いられるデータを示す図である。

【図１４】

第２の実施の形態に係る各ノード間のコマンドやデータの授受を説明するダイアグラムを示す図である。

【図１５】

第２の実施の形態においてデスティネーションに対して設定を行うコマンドパケットに用いられるデータを示す図である。

【図１６】

第２の実施の形態においてソースに対して設定を行うコマンドパケットに用いられるデータを示す図である。

【図１７】

第３の実施の形態のオブジェクトデータの転送を説明するための図である。

【図１８】本発明の第３の実施の形態に係る各ノード間のコマンドやデータの授受を説明するダイアグラムを示す図である。

【図１９】

第１の実施の形態の情報を管理用テーブルの説明図である。

【図２０】

第２の実施の形態の情報を管理用テーブルの説明図である。

【図２１】

従来例の説明図である。

【符号の説明】

10　computer

12　演算処理装置(MPU)

14　第一の1394インターフェイス

16　キーボードなど第一の操作部

18　第一のデコーダ

20　CRT ディスプレイなどの表示装置

22　ハードディスク

24　第一のメモリ

26　PCI バスなどのコンピュータ内部バス

28　VCR

30　撮像光学系

32　A/D 変換器

34　ビデオ処理部

36　圧縮伸長回路

38　第一のメモリ

40　第二のメモリ

42　第一のデータセレクタ

44　第二の1394インターフェイス

46　第一のメモリ制御回路

48　第二のメモリ制御回路

50　システムコントローラ

52　第二の操作部

54　電子ビューファインダ

56　D/A 変換器

58　記録部

60　プリンタ

62　第三の1394インターフェイス

64　第二のデータセレクタ

66　第三の操作部

68　プリンタコントローラ

70　第二のデコーダ

72　第三のメモリ

74　画像処理部

76　ドライバ

78　プリンタヘッド

200 コントロールノード

202 ソースノード

204 デスティネーションノード

206 ソースノード内部のサブユニット

208 画像データ等のobject

210 デスティネーションノード内部の第一のメモリ空間

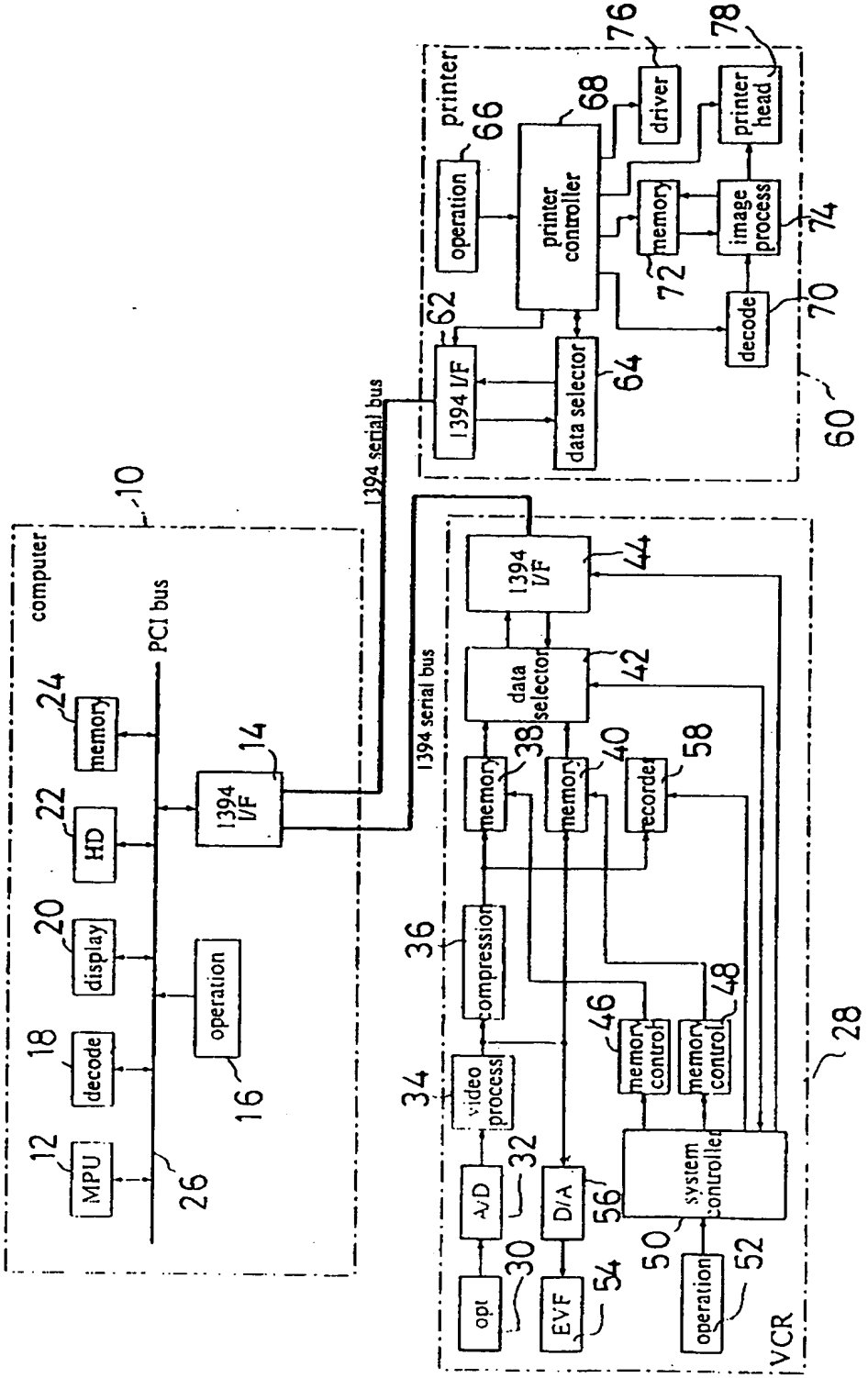
212 第一のコネクション

214 デスティネーションノード内部の第n のメモリ空間

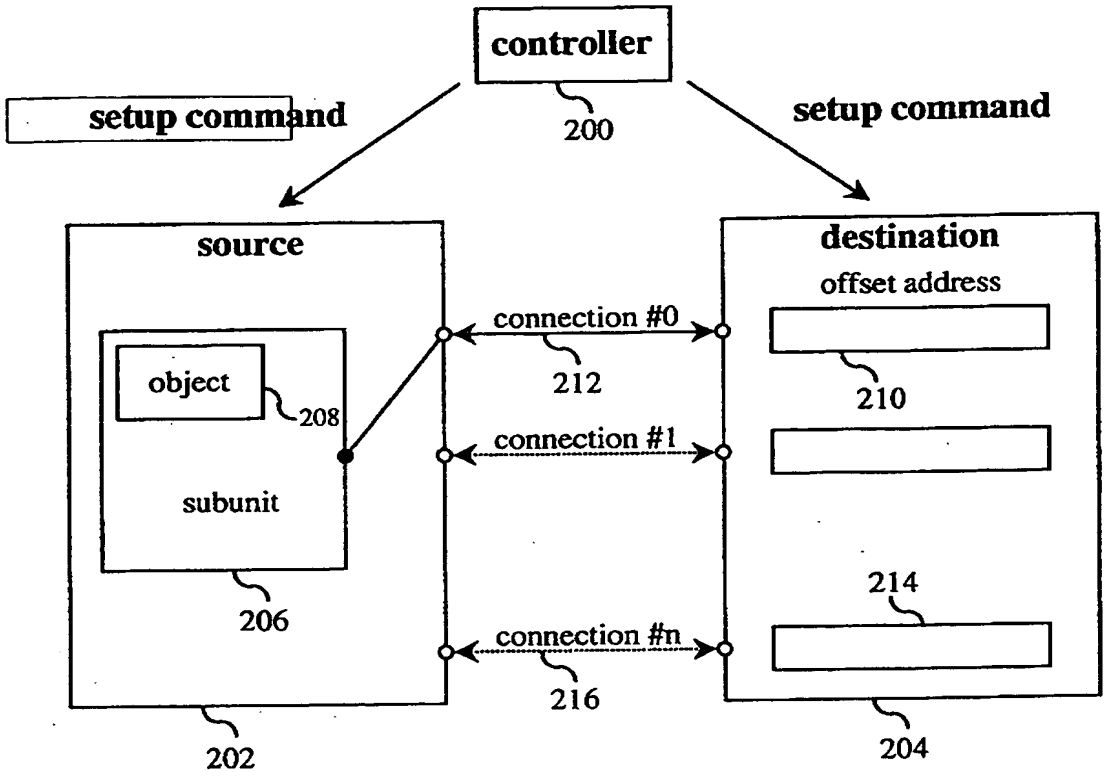
216 第ｎのコネクション

【書類名】　図面

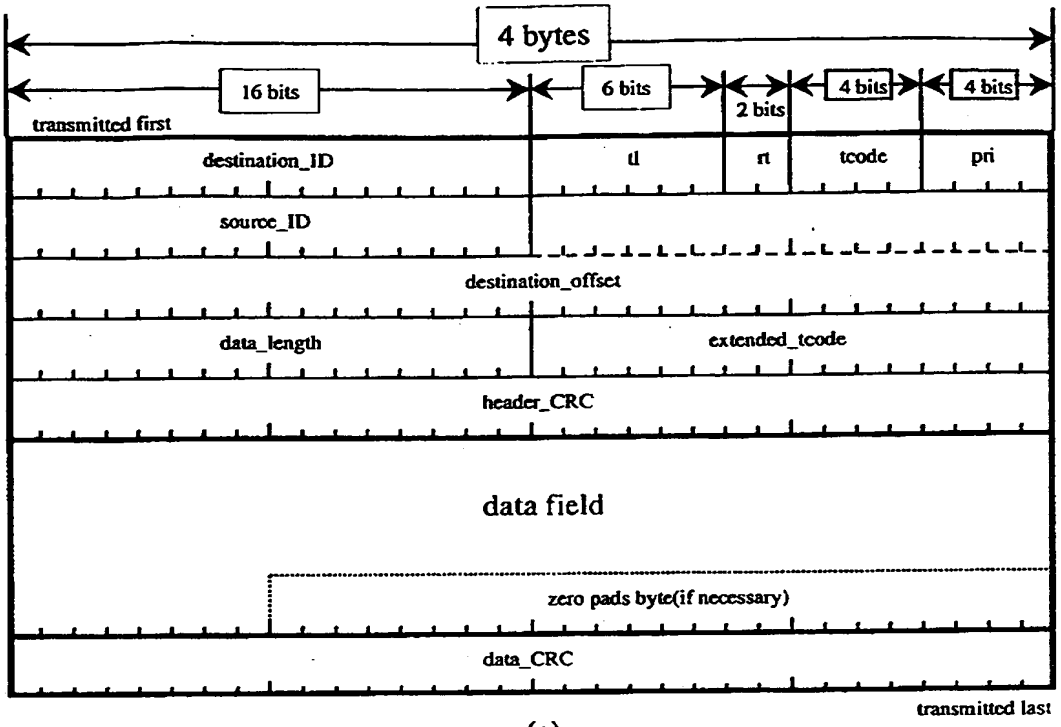
【図１】


computer
10
PCI bus
MPU
12
decode
18
display
20
HD
22
memory
24
operation
16
26
1394 I/F
14
1394 serial bus
printer
60
1394 I/F
62
operation
66
printer controller
68
data selector
64
memory
72
image process
74
decode
70
driver
76
printer head
78
VCR
28
1394 I/F
44
data selector
42
memory
38
memory
40
recorder
58
compression
36
video process
34
A/D
32
opt
30
D/A
56
EVF
54
memory control
46
memory control
48
system controller
50
operation
52

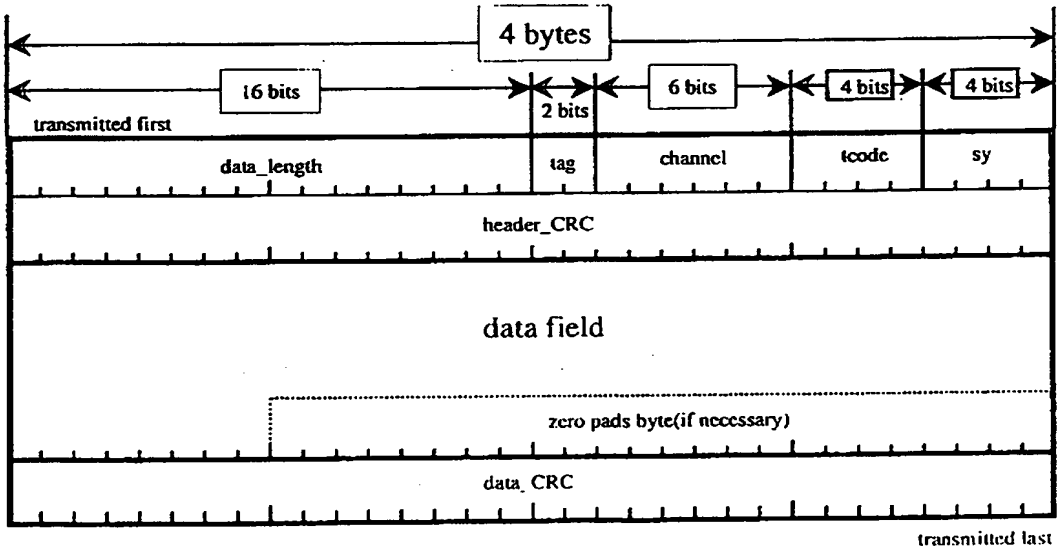
【図２】


controller
200
setup command
setup command
source
destination
offset address
connection #0
212
object
208
210
connection #1
subunit
214
206
connection #n
216
202
204

【図３】

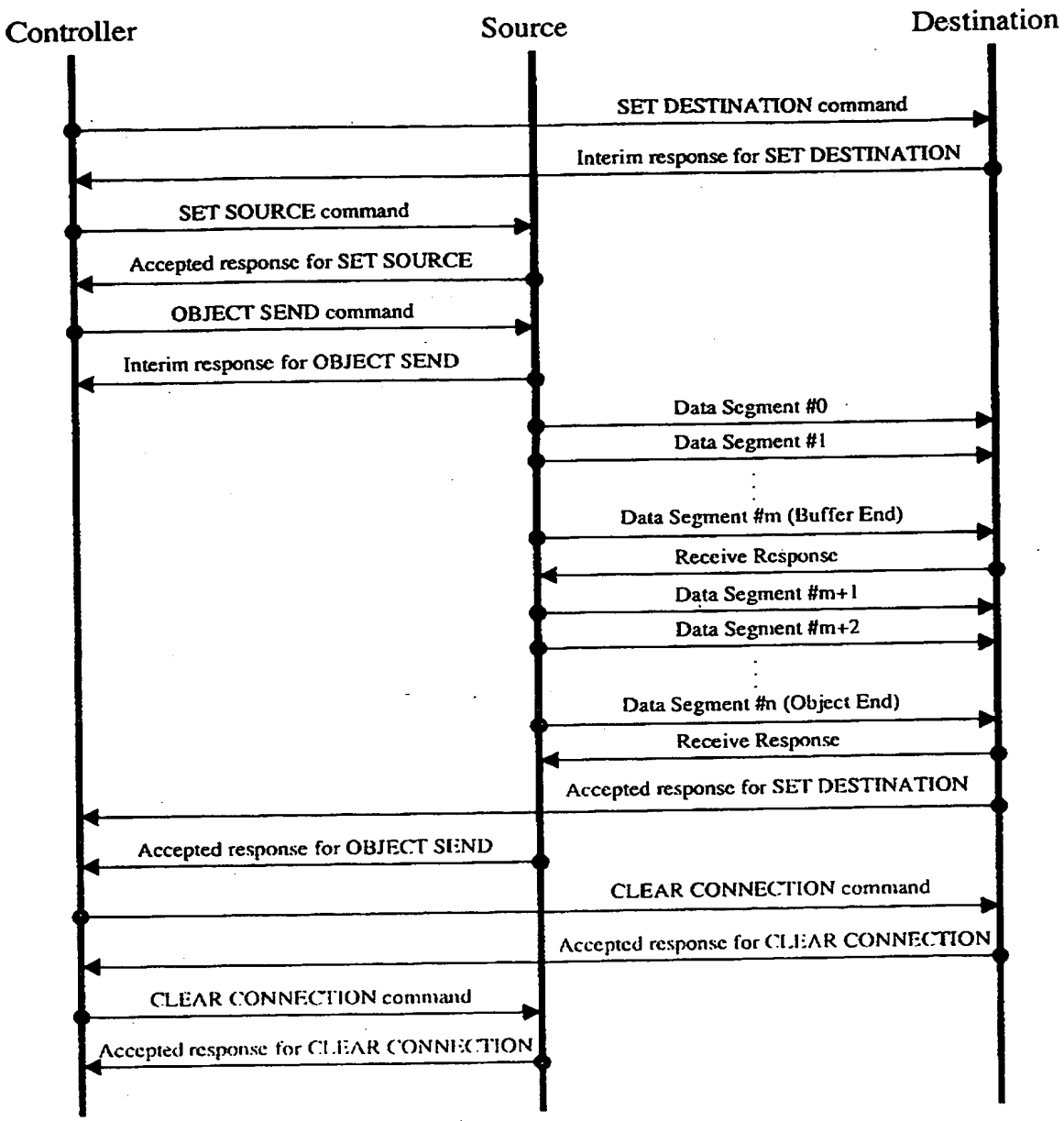

4 bytes
16 bits
6 bits
2 bits
4 bits
4 bits
transmitted first
destination_ID
tl
rt
tcode
pri
source_ID
destination_offset
data_length
extended_tcode
header_CRC
data field
zero pads byte(if necessary)
data_CRC
transmitted last

(a)

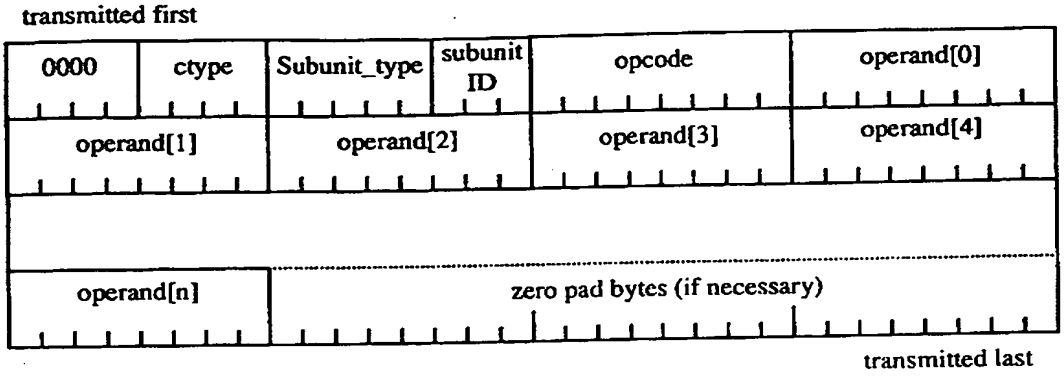

4 bytes
16 bits
2 bits
6 bits
4 bits
4 bits
transmitted first
data_length
tag
channel
tcode
sy
header_CRC
data field
zero pads byte(if necessary)
data_CRC
transmitted last

(b)

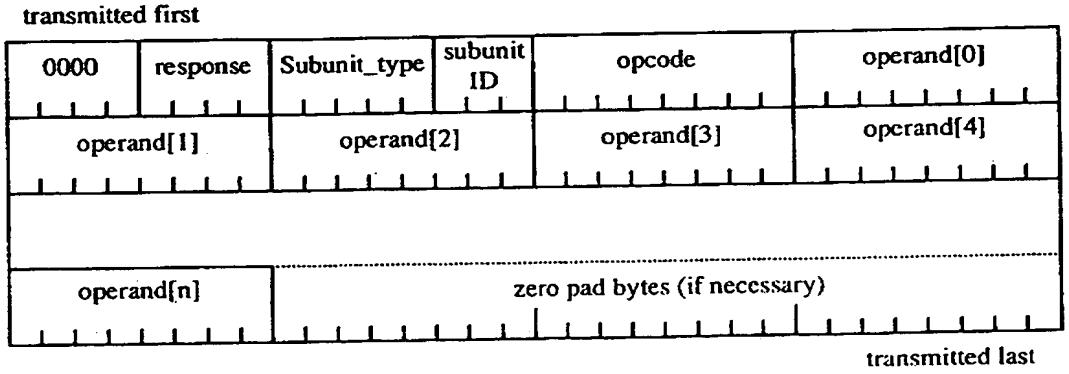
【図４】


Controller
Source
Destination
SET DESTINATION command
Interim response for SET DESTINATION
SET SOURCE command
Accepted response for SET SOURCE
OBJECT SEND command
Interim response for OBJECT SEND
Data Segment #0
Data Segment #1
Data Segment #m (Buffer End)
Receive Response
Data Segment #m+1
Data Segment #m+2
Data Segment #n (Object End)
Receive Response
Accepted response for SET DESTINATION
Accepted response for OBJECT SEND
CLEAR CONNECTION command
Accepted response for CLEAR CONNECTION
CLEAR CONNECTION command
Accepted response for CLEAR CONNECTION

【図５】


transmitted first
0000
ctype
Subunit_type
subunit ID
opcode
operand[0]
operand[1]
operand[2]
operand[3]
operand[4]
operand[n]
zero pad bytes (if necessary)
transmitted last

【図６】


transmitted first
0000
response
Subunit_type
subunit ID
opcode
operand[0]
operand[1]
operand[2]
operand[3]
operand[4]
operand[n]
zero pad bytes (if necessary)
transmitted last

【図７】

	msb							lsb
opcode	SET DESTINATION							
operand[0]	node_vendor_id							
operand[1]								
operand[2]								
operand[3]	chip_id_hi							
operand[4]	chip_id_lo							
operand[5]								
operand[6]								
operand[7]								
operand[8]	connection_id							
operand[9]	reserved						max_rec	
operand[10]	buffer_size							
operand[11]								
operand[12]								
operand[13]	status_info							

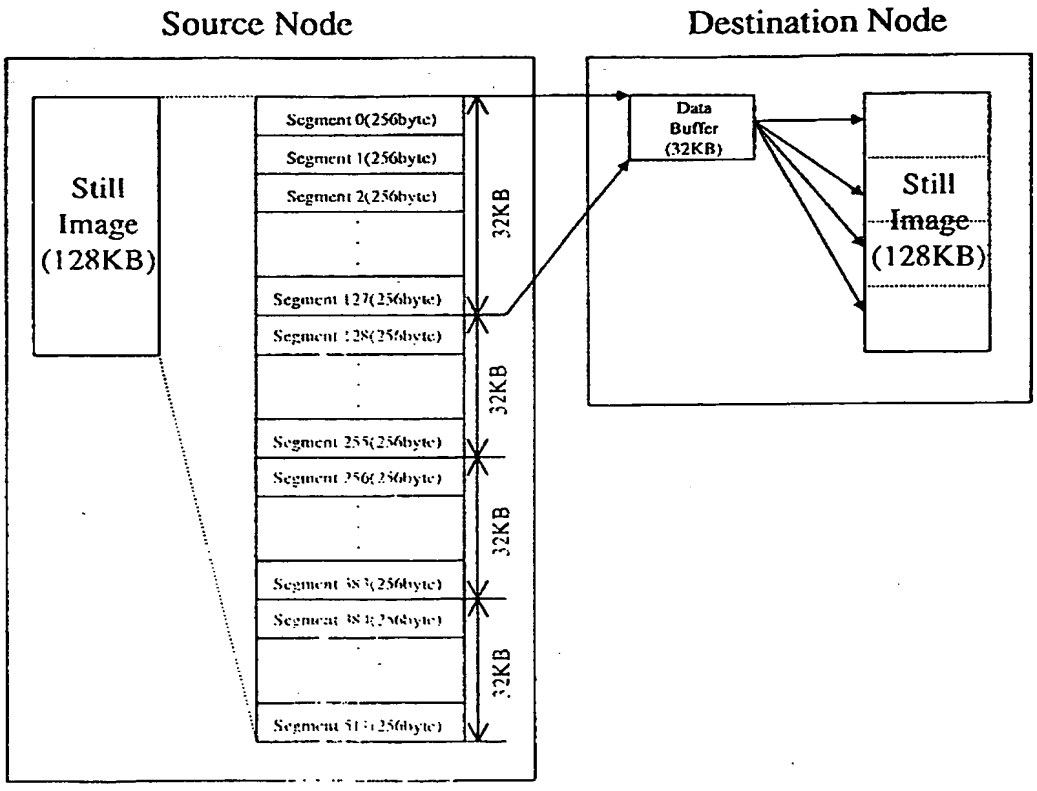
【図８】

	msb							lsb
opcode	SET SOURCE							
operand[0]	node_vendor_id							
operand[1]								
operand[2]								
operand[3]	chip_id_hi							
operand[4]	chip_id_lo							
operand[5]								
operand[6]								
operand[7]								
operand[8]	connection_id							
operand[9]	reserved						max_rec	
operand[10]	buffer_size							
operand[11]								
operand[12]								
operand[13]	number_of_destinations							
operand[14]	status_info							

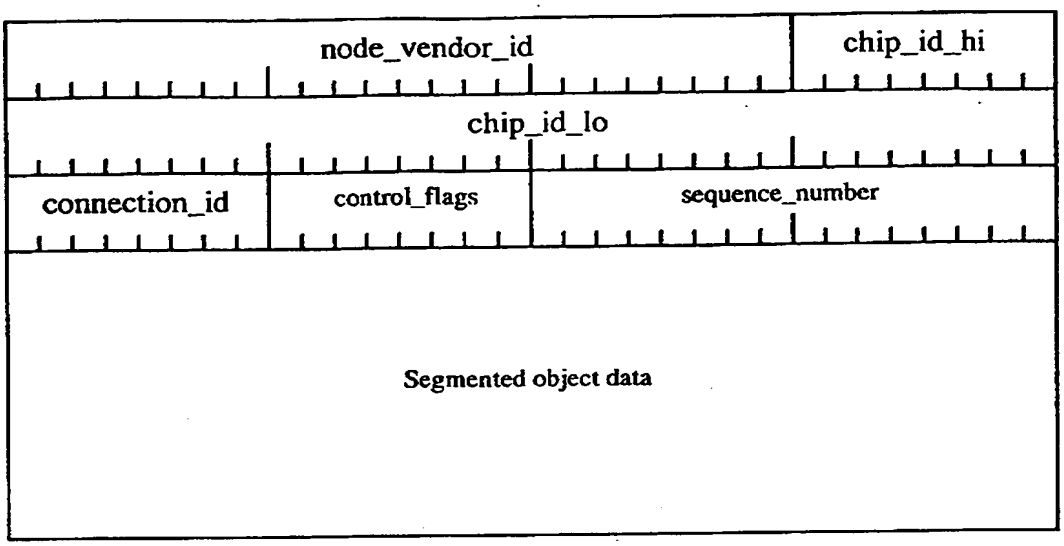
【図９】

	msb							lsb
opcode	OBJECT SEND							
operand[0]	node_vendor_id							
operand[1]								
operand[2]								
operand[3]	chip_id_hi							
operand[4]	chip_id_lo							
operand[5]								
operand[6]								
operand[7]								
operand[8]	connection_id							
operand[9]	subfunction							
operand[10]	status_info							

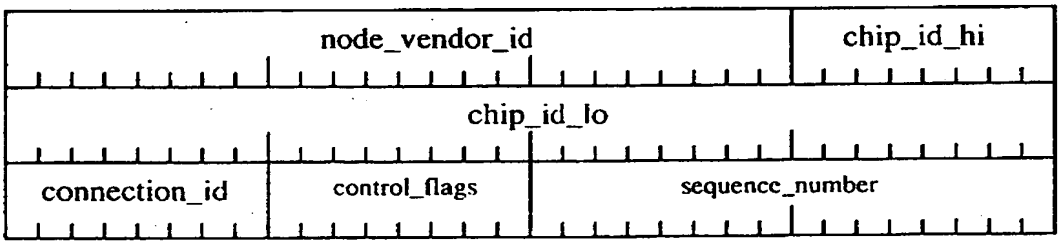
【図１０】



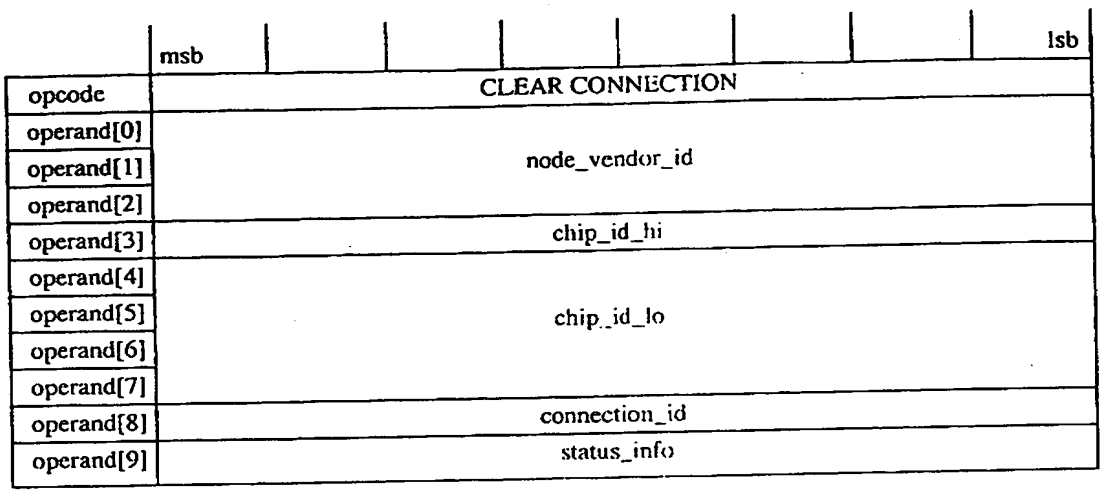
【図１１】


node_vendor_id
chip_id_hi
chip_id_lo
connection_id
control_flags
sequence_number
Segmented object data

【図１２】

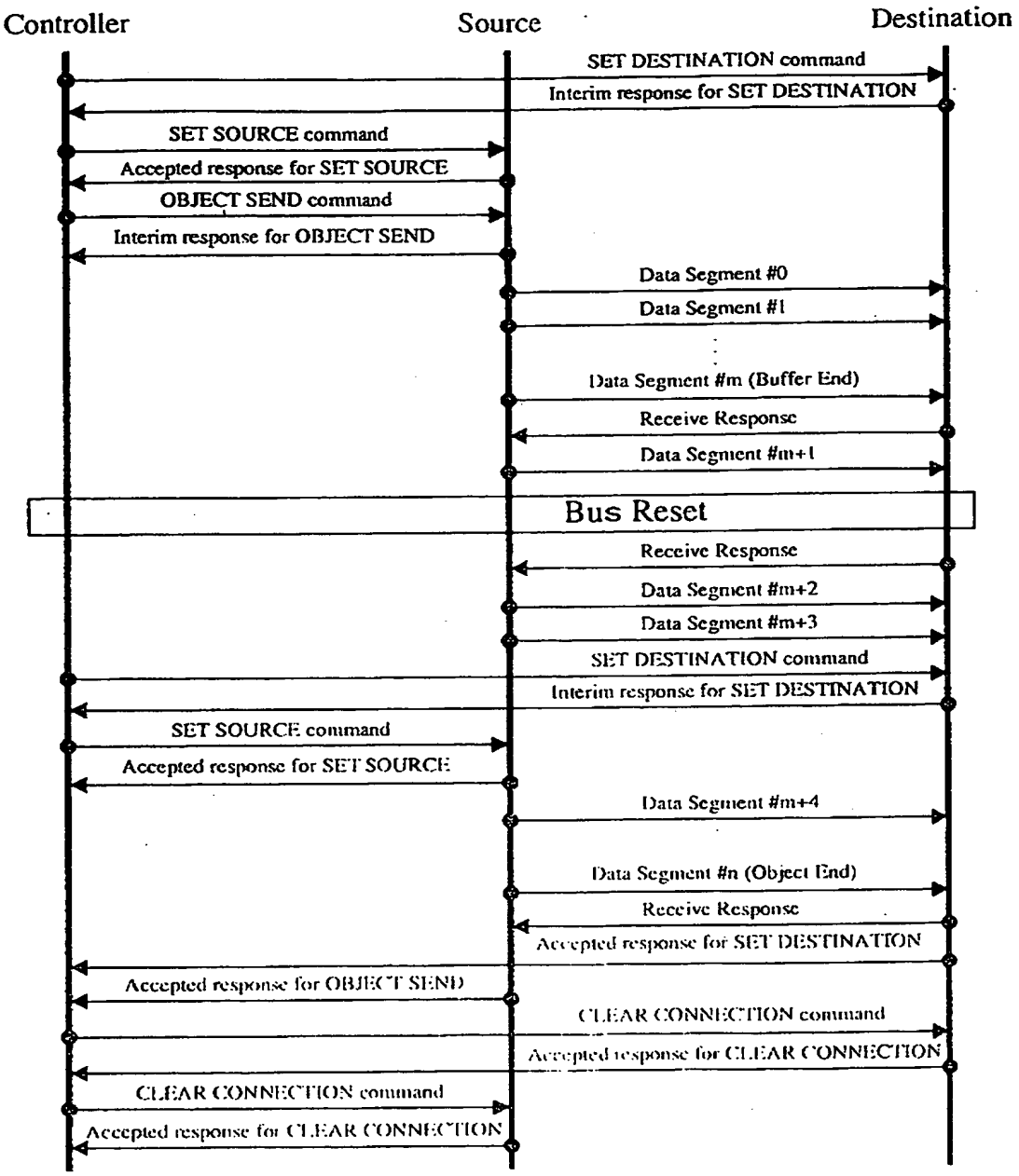

node_vendor_id
chip_id_hi
chip_id_lo
connection_id
control_flags
sequence_number

【図１３】

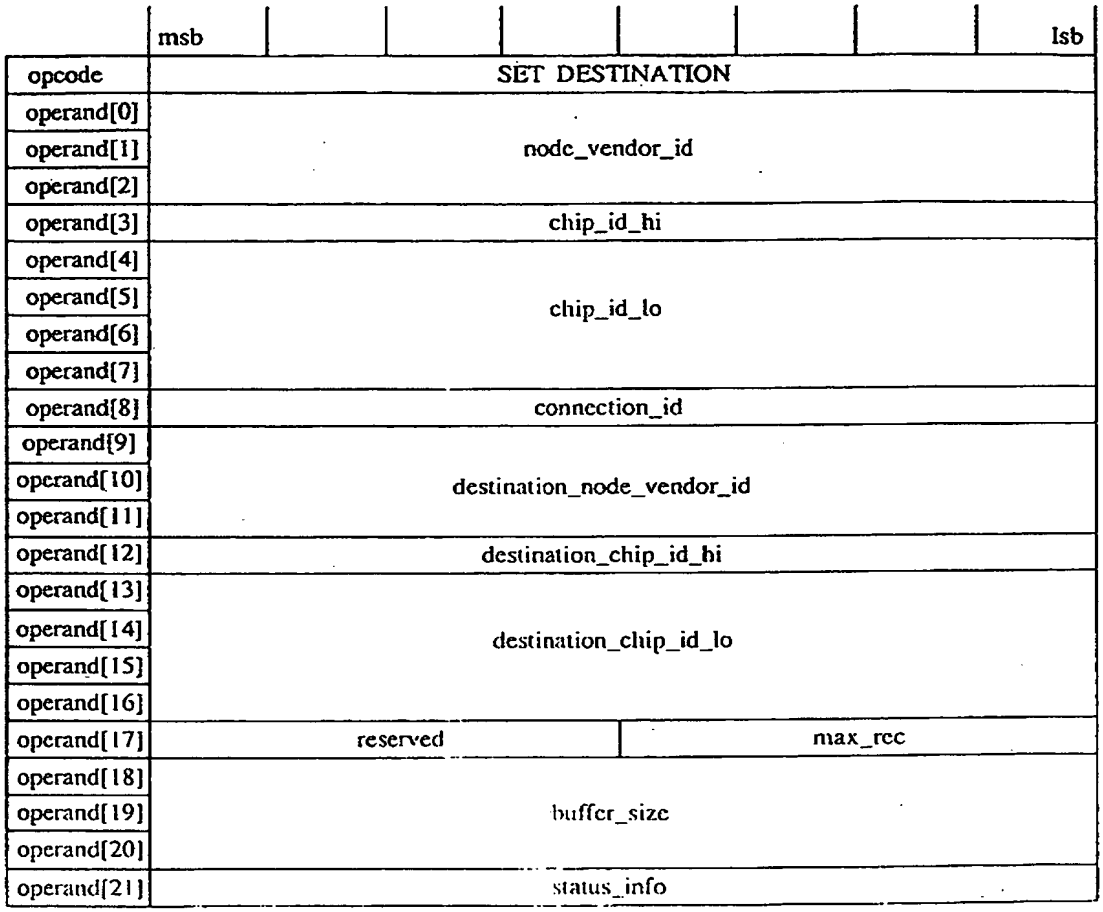


	msb							lsb
opcode	CLEAR CONNECTION							
operand[0]	node_vendor_id							
operand[1]								
operand[2]								
operand[3]	chip_id_hi							
operand[4]	chip_id_lo							
operand[5]								
operand[6]								
operand[7]								
operand[8]	connection_id							
operand[9]	status_info							

【図１４】

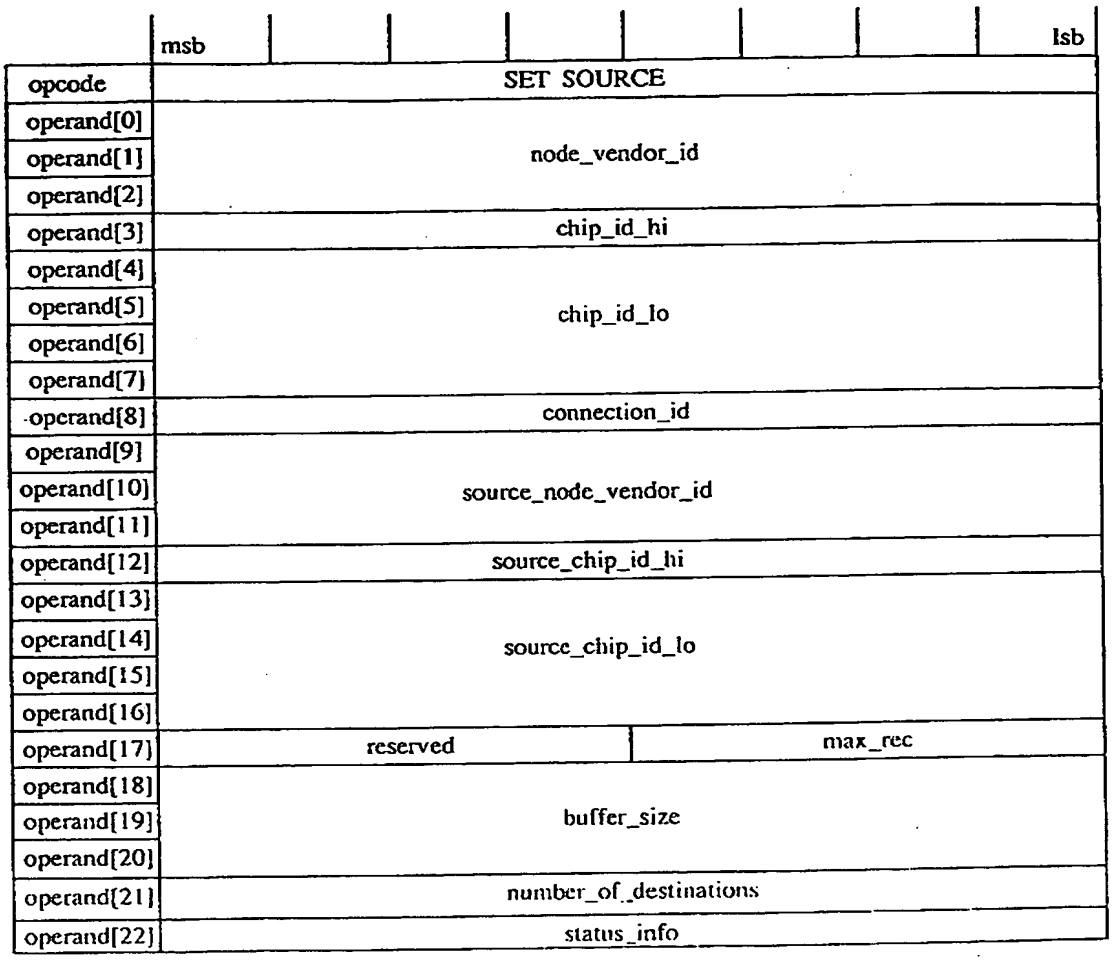

Controller
Source
Destination
SET DESTINATION command
Interim response for SET DESTINATION
SET SOURCE command
Accepted response for SET SOURCE
OBJECT SEND command
Interim response for OBJECT SEND
Data Segment #0
Data Segment #1
Data Segment #m (Buffer End)
Receive Response
Data Segment #m+1
Bus Reset
Receive Response
Data Segment #m+2
Data Segment #m+3
SET DESTINATION command
Interim response for SET DESTINATION
SET SOURCE command
Accepted response for SET SOURCE
Data Segment #m+4
Data Segment #n (Object End)
Receive Response
Accepted response for SET DESTINATION
Accepted response for OBJECT SEND
CLEAR CONNECTION command
Accepted response for CLEAR CONNECTION
CLEAR CONNECTION command
Accepted response for CLEAR CONNECTION

【図１５】



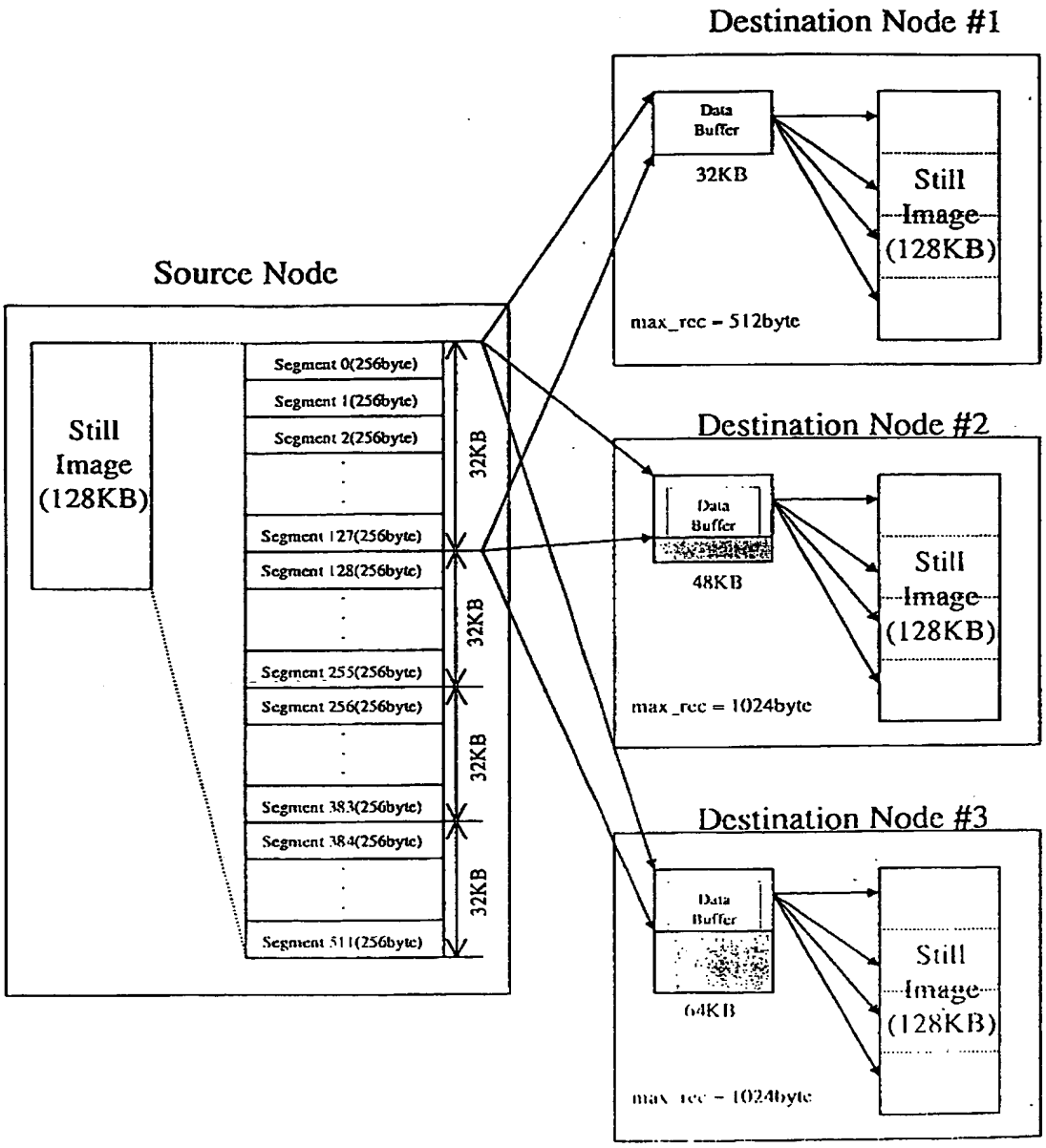
	msb						lsb
opcode	SET DESTINATION						
operand[0]	node_vendor_id						
operand[1]							
operand[2]							
operand[3]	chip_id_hi						
operand[4]	chip_id_lo						
operand[5]							
operand[6]							
operand[7]							
operand[8]	connection_id						
operand[9]	destination_node_vendor_id						
operand[10]							
operand[11]							
operand[12]	destination_chip_id_hi						
operand[13]	destination_chip_id_lo						
operand[14]							
operand[15]							
operand[16]							
operand[17]	reserved					max_rec	
operand[18]	buffer_size						
operand[19]							
operand[20]							
operand[21]	status_info						

【図１６】

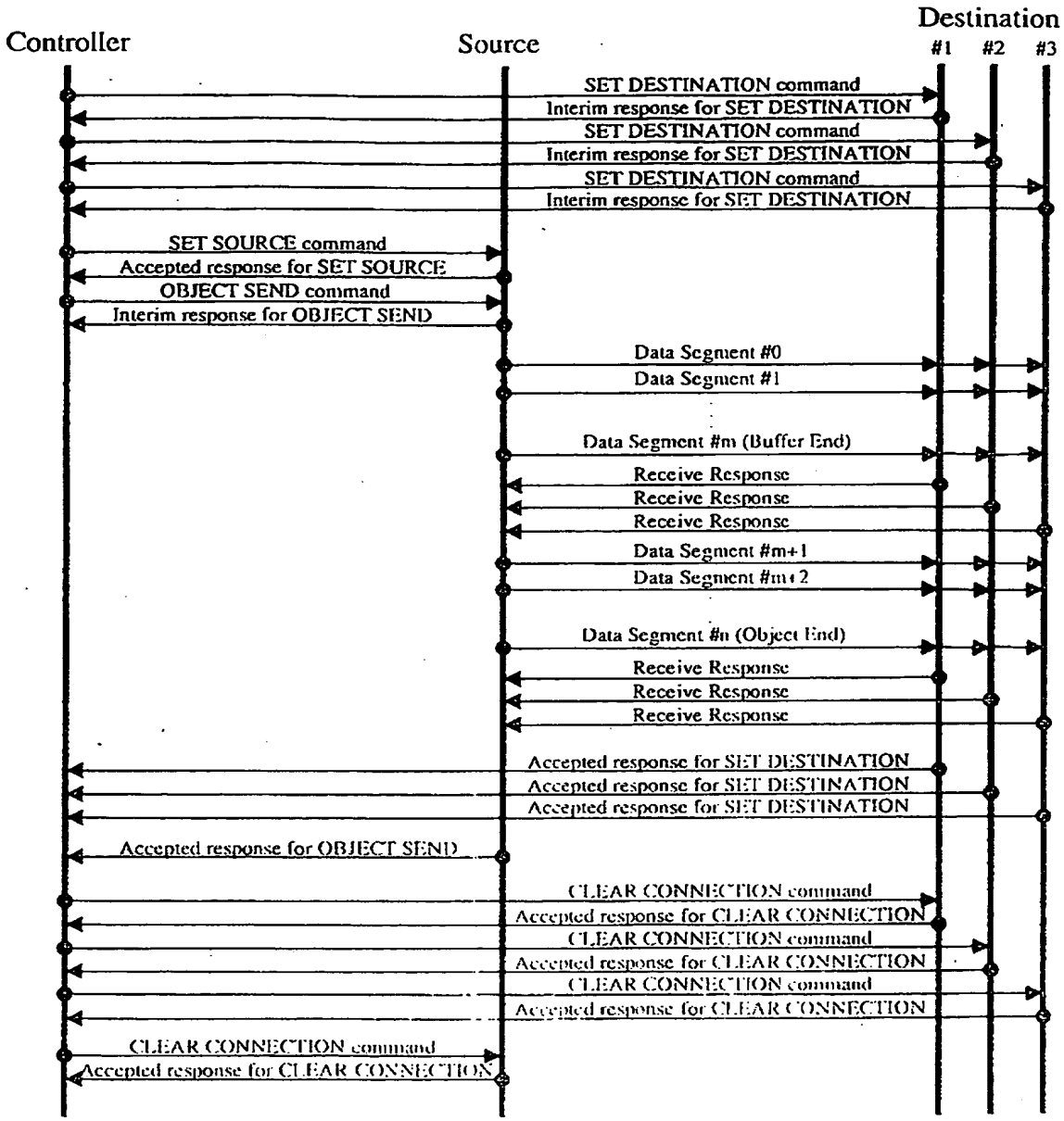


	msb							lsb
opcode	SET SOURCE							
operand[0]	node_vendor_id							
operand[1]								
operand[2]								
operand[3]	chip_id_hi							
operand[4]	chip_id_lo							
operand[5]								
operand[6]								
operand[7]								
operand[8]	connection_id							
operand[9]	source_node_vendor_id							
operand[10]								
operand[11]								
operand[12]	source_chip_id_hi							
operand[13]	source_chip_id_lo							
operand[14]								
operand[15]								
operand[16]								
operand[17]	reserved						max_rec	
operand[18]	buffer_size							
operand[19]								
operand[20]								
operand[21]	number_of_destinations							
operand[22]	status_info							

【図１７】


Destination Node #1
Data Buffer
32KB
Still Image (128KB)
max_rec = 512byte
Source Node
Still Image (128KB)
Segment 0(256byte)
Segment 1(256byte)
Segment 2(256byte)
Segment 127(256byte)
Segment 128(256byte)
Segment 255(256byte)
Segment 256(256byte)
Segment 383(256byte)
Segment 384(256byte)
Segment 511(256byte)
32KB
32KB
32KB
32KB
Destination Node #2
Data Buffer
48KB
Still Image (128KB)
max_rec = 1024byte
Destination Node #3
Data Buffer
64KB
Still Image (128KB)

【図１８】


Controller
Source
Destination
#1
#2
#3
SET DESTINATION command
Interim response for SET DESTINATION
SET DESTINATION command
Interim response for SET DESTINATION
SET DESTINATION command
Interim response for SET DESTINATION
SET SOURCE command
Accepted response for SET SOURCE
OBJECT SEND command
Interim response for OBJECT SEND
Data Segment #0
Data Segment #1
Data Segment #m (Buffer End)
Receive Response
Receive Response
Receive Response
Data Segment #m+1
Data Segment #m+2
Data Segment #n (Object End)
Receive Response
Receive Response
Receive Response
Accepted response for SET DESTINATION
Accepted response for SET DESTINATION
Accepted response for SET DESTINATION
Accepted response for OBJECT SEND
CLEAR CONNECTION command
Accepted response for CLEAR CONNECTION
CLEAR CONNECTION command
Accepted response for CLEAR CONNECTION
CLEAR CONNECTION command
Accepted response for CLEAR CONNECTION
CLEAR CONNECTION command
Accepted response for CLEAR CONNECTION

【図１９】

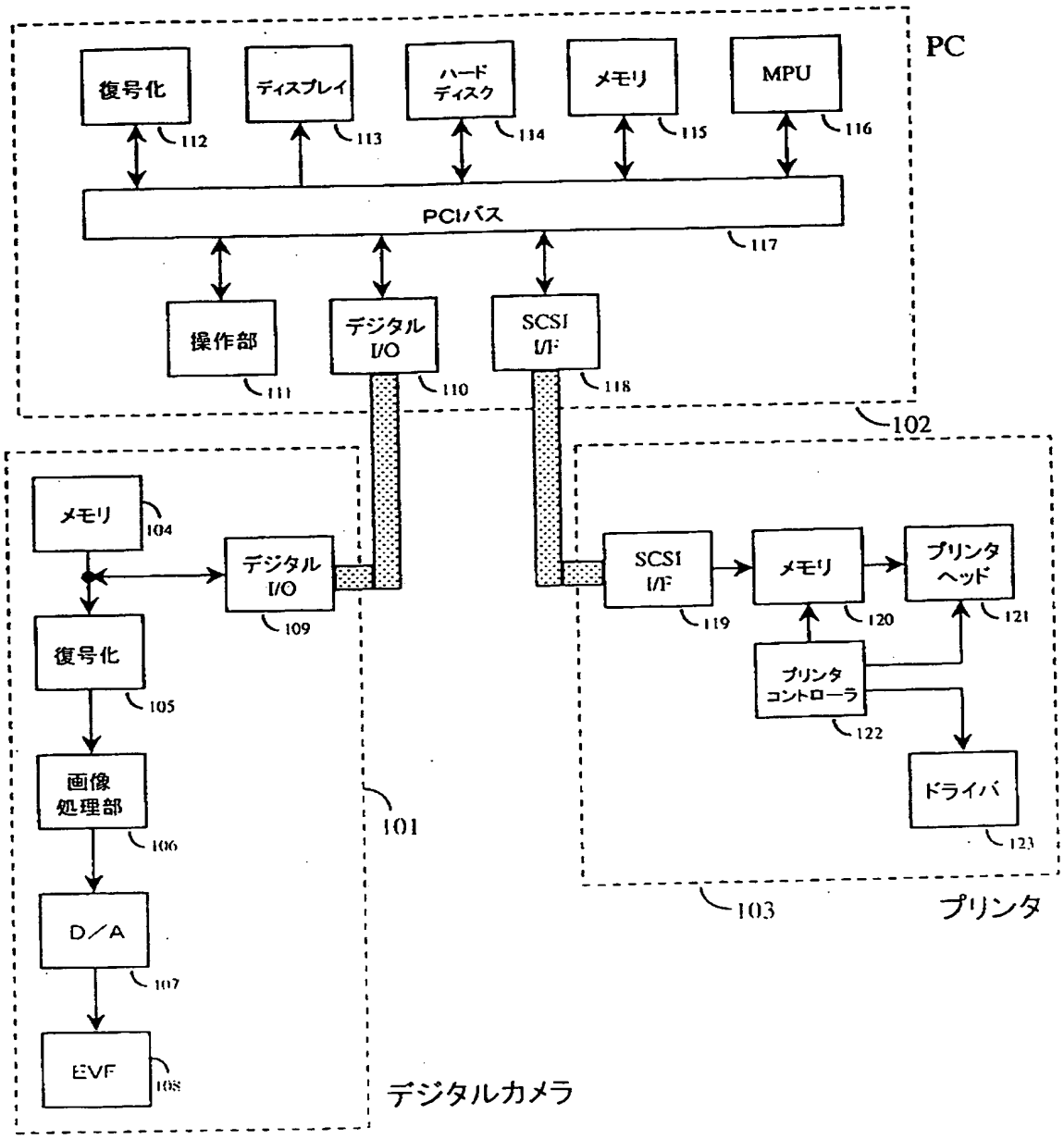
connection_id	flag	max_rec	buffer_size
0	1	512	32KB
1	0	0	0
		:	
n-1	0	0	0
n	0	0	0

【図２０】

connection_id	flag	max_rec	buffer_size	number_of_destinations
0	1	512	32KB	1
1	0	0	0	0
			:	
n-1	0	0	0	0
n	0	0	0	0

EUI-64	
Source	xxxxxx xx xxxxxxxx
Destination #1	yyyyyy yy yyyyyyyy

【図２１】


PC
復号化
112
ディスプレイ
113
ハード
ディスク
114
メモリ
115
MPU
116
PCIバス
117
操作部
111
デジタル
I/O
110
SCSI
I/F
118
102
メモリ
104
デジタル
I/O
109
復号化
105
画像
処理部
106
D／A
107
EVF
108
101
デジタルカメラ
SCSI
I/F
119
メモリ
120
プリンタ
ヘッド
121
プリンタ
コントローラ
122
ドライバ
123
103
プリンタ

【書類名】　　要約書

【要約】

【課題】　簡便にかつ高速にデータを転送できるようにするとともに、確実にデータ転送を実行できるようにする。

【解決手段】　所定の機器に固有の固有情報と、送信機器と１つ以上の受信機器との間の論理的な接続を示すコネクション情報とを用いて、情報データの送受信を行うデータ通信システムにおいて、上記１つ以上の受信機器の受信能力のうち、最も低い受信能力に従って上記情報データの送受信を行うようにすることにより、各デスティネーションに設けられているバッファサイズに関係なく、ソースは通知されたバッファサイズ以下のサイズ単位でデータ転送を行い、通信を確実に実行することが可能となる。

【選択図】　　図１

【書類名】　職権訂正データ
【訂正書類】　特許願

＜認定情報・付加情報＞
【特許出願人】
【識別番号】　000001007
【住所又は居所】　東京都大田区下丸子３丁目３０番２号
【氏名又は名称】　キヤノン株式会社
【代理人】　申請人
【識別番号】　100090273
【住所又は居所】　東京都豊島区東池袋１丁目１７番８号　池袋ＴＧホーメストビル５階　國分特許事務所
【氏名又は名称】　國分　孝悦

出証特平１１－３０１６２４８

出　願　人　履　歴　情　報

識別番号　　〔０００００１００７〕

１．変更年月日　　１９９０年　８月３０日
　〔変更理由〕　　新規登録
　　住　所　　東京都大田区下丸子３丁目３０番２号
　　氏　名　　キヤノン株式会社